Soft Computing
https://doi.org/10.1007/s00500-018-3561-7

METHODOLOGIES AND APPLICATION

An Evolutionary Algorithm and operators for the Airport Baggage Sorting Station Problem

Amadeo Ascó[1]

Abstract
Taking into consideration the constraints and objectives to appropriately assigning the available airport resources throughout the period of time an airport provides its services can greatly affect the quality of service which airlines and airports provide to their customers. The appropriate assignments can help airlines and airports to keep to published schedules, by minimising changes in these schedules, reducing delays and considering customers preferences when assigning the resources. Given the expected increases in civil air traffic, and the variety of resources, the complexities of resource scheduling and assignment continue to increase. For this reason, as well as the dynamic nature of the problems, scheduling and assignment are becoming increasingly more difficult. An Evolutionary Algorithm is presented together with some different operators, which are used to find good solutions to the Airport Baggage Sorting Station Assignment Problem for when there are not sufficient resources up to when the number of resources is sufficient to fulfil the demand on these resources. The contributions of these different operators are studied and compared to other approaches, giving insights into how the appropriate choice may depend upon the specifics of the problem at the time.

Keywords Airport Baggage Sorting Station Problem · Scheduling · Heuristics · Evolutionary Algorithms

1 Overview

Many airport resources (for example, stands, gates, tugs, storage points, fuel trucks and baggage stations) are of limited availability and are expensive or time-consuming to increase in quantity. Given this, airports need to use their resources as efficiently as possible, since any delays due to lack of available resources can have a direct impact upon the published schedules, the provision of services to customers and the workforce. Moreover, the characteristics of each problem are represented by constraints and the desired solutions by objectives, which naturally conflict with each other.

The problem can be represented by some constraints which must be strictly complied with (known as hard constraints) and other constraints where compliance is desirable (called soft constraints or objectives). In order to solve a problem, it is necessary to find solutions which comply with both the hard constraints and most or all of the soft constraints. An indication of the compliance with the soft constraints is provided by an evaluation function, sometimes referred to as the fitness function, the results of which give an indication as to the quality or fitness of the solutions.

Many approaches have been used to solve optimisation problems, like the Airport Baggage Sorting Station Assignment Problem (ABSSAP), two of which are Genetic Algorithms (GAs) and Tabu Search (TS). GAs are one of the methodologies belonging to the population-based model of Evolutionary Algorithms (EAs), based on the Darwin and Wallace (1858) theory of natural selection and Mendelian genetics (Mendel 1865), which are recognised as the foundation of evolutionary biology. GAs have been used to solve a wide range of airport problems, such as the Airport Gate Assignment Problem (AGAP) in Lim et al. (2005), the scheduling of arriving aircraft in Cheng et al. (1999), Xiangwei et al. (2010) and Hansen (2004), the scheduling of departing aircraft in Bolender (2000) and Caprì and Ignaccolo (2004), the aircraft taxiing in Gotteland and Durand (2003) and the ABSSAP in Ascó (2016) and Ascó et al. (2010, 2012, 2013). TS is a Metaheuristic algorithm which employs a local search, which in turn uses a solution to generate a neighbour-

Communicated by V. Loia.

✉ Amadeo Ascó
a.asco@bocaditos.co.uk

1 University of Nottingham, Sandbach, Moston, UK

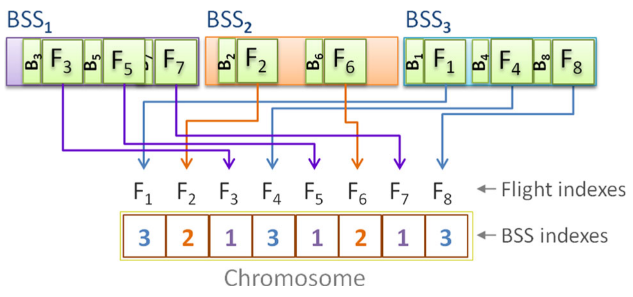

Fig. 1 An example of encoding for a three BSSs and eight flights

hood of solutions. The solutions from the neighbourhood are checked in the hope of finding an improved solution. A local search may get stuck within areas of the search space where the neighbourhood is equally fit, so memory structures which describe the neighbourhood visited are incorporated to avoid using again solutions and regions previously visited, Glover (1989, 1990) and Burke and Kendall (2005). An implementation of both a GA and TS is used later in this paper, where the neighbourhood (also called local walk) is generated by using the mutation operators described in Sect. 6, which constitute the list of candidate solutions. In the TS, the fittest non-tabu solution in the candidate list is adopted as the new current solution and is also added to the tabu list. Once the tabu list is full, one solution is removed from the tabu list to leave space for the new tabu solution.

Various different ABSSAP objectives have to be considered, such as maximising assignments, ensuring full service time and allocating preferential positions. Some of these objectives are in obvious conflict (reducing service times in order to service an additional flight, for example), thus preventing simultaneous optimisation of each objective.

An encoding of the parameter set for the ABSSAP for the Canonical Genetic Algorithm (CGA) was implemented using the Evolutionary Computation Java library (ECJ), where a chromosome is composed of the indexes of the baggage sorting station (BSS) assigned to each flight, the flights being ordered by their base service starting time, as shown in Fig. 1.

The initial studies showed that good initial solutions greatly improve the speed, convergence and quality of the final solutions to the limited time ranges under consideration, as shown in Sect. 8.

The following sections begin by a description of the problem, followed by a description of the proposed EA with its operators and selectors, followed by a study of the problem, using a fitness function as a single compound objective which represents realistic priorities.

2 The Airport Baggage Sorting Station Assignment Problem

The checked-in baggage at a passenger airport first enters the baggage system where it is processed and delivered to the ground side, and an overview of the process is provided in Fig. 2. The baggage is then transported by conveyor belts to the baggage system's security hall where it is individually scanned. Most baggage will continue straight on, but if at the scanning stage suspicions were aroused concerning the baggage, then it is diverted to the security checking area where it will be further checked by one of the security personnel and, if clear, will rejoin the normal journey with the rest of the baggage. In the area of conveyor systems, Johnstone et al. (2015) investigate the design and control of merging bottlenecks of conveyor-based baggage handling systems, and Kim et al. (2017) looks at determining an appropriate workload balance for a Baggage Handling System (BHS). The baggage will then continue (on conveyor belts) to the baggage hall and be transported to the baggage sorting station assigned to it. Once the baggage reaches the BSS, it accumulates ready for the workforce to sort and place on trolleys or into special containers, which go directly into the aircraft, ready for transportation by cart and placed next to the aircraft on the air side where the baggage is loaded into the aircraft hold by the ground workforce, ready to travel to its destination. Containers are used to transport the baggage on wide fuselage aircraft, for long distance flights, which are directly placed into the hold of the aircraft. The proposed EA here assigns these BSSs to the aircraft (flights) under the described constraints and objectives presented in Sect. 3. Trolleys are used in the narrow fuselage aircraft, and the baggage is individually loaded into the hold of the aircraft by the handling workforce, who use conveyor belts to lift the baggage from the trolleys to the level of the aircraft's hold. Johnstone et al. (2010) looked at the routing of the baggage within the baggage system with the aim of providing additional insight into how agents can learn to route in a baggage handling system, and experiments show that the learning method performs better than the search method.

On reaching the destination airport, the process is reversed, so that the ground workforce removes the baggage from the cargo area of the aircraft and places it directly on baggage carts (open trolleys, onto which baggage is separately loaded and protected with a canvas cover) or loads in baggage containers onto dollies (trailers, on which baggage containers are loaded) ready for transportation by cart to the baggage sorting stations assigned. Here the handling force transfers it from the trolleys or containers onto the baggage sorting stations for transportation to the ground side of the arrival hall. The baggage then enters the baggage system which delivers the baggage to the carousel to which the flight is assigned, in readiness for collection by the corresponding owner, and will then leave the airport. In the case of transfer passengers, their baggage is delivered to the baggage sorting station assigned to their next flight. The sorting station used by a flight arrival is normally directly linked to the carousel assigned to the passenger flight for the given destination, and only the trans-

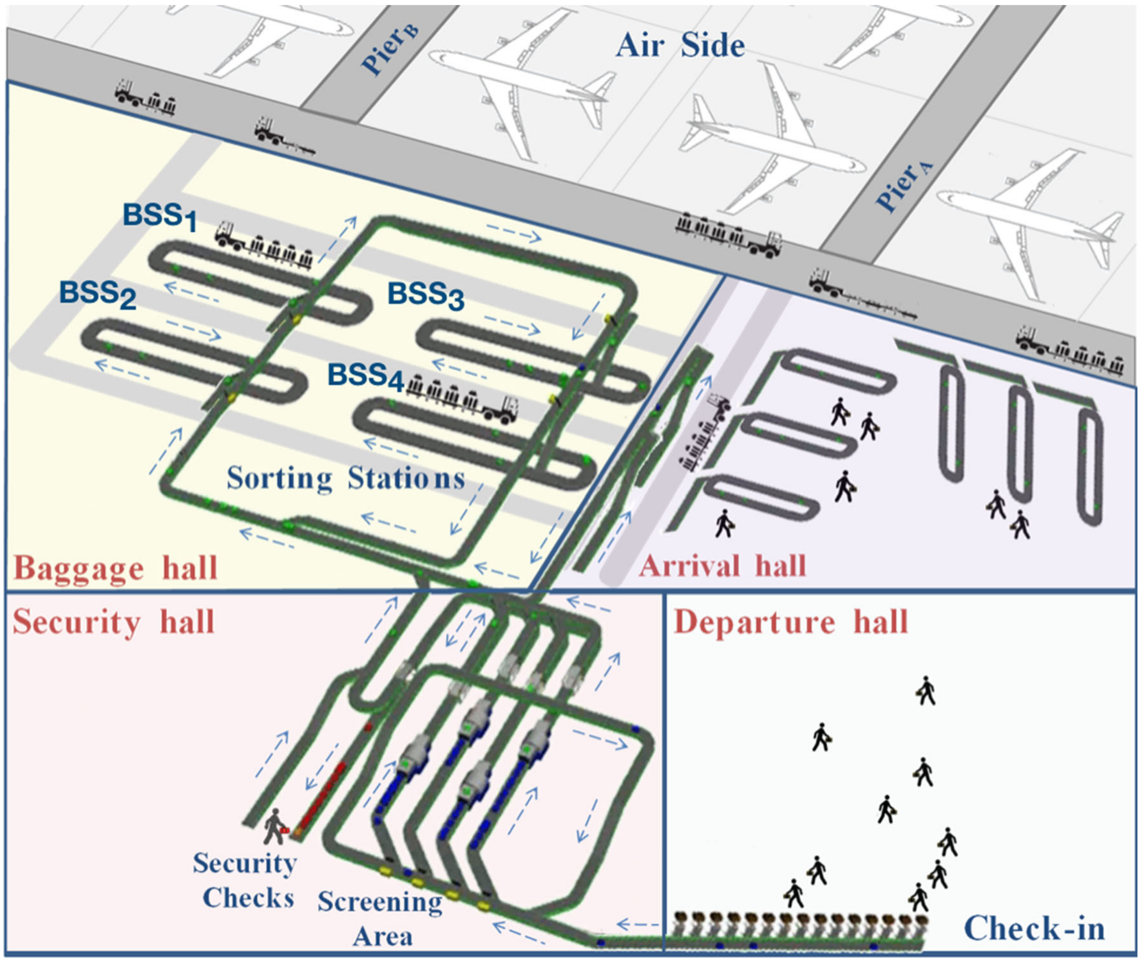

Fig. 2 Simplified overview of baggage system

fer baggage re-enters the baggage system for delivery to the sorting station assigned to its next departure, as shown in the ‘Arrival hall’ in Fig. 2. The transfer baggage does not usually need to be directed through the security hall, given that it should already have been checked at the original airport.

Where airports have several terminals, it would be unrealistic to assume that baggage from a flight at a terminal stand is serviced by a baggage sorting station in another terminal (e.g. passengers usually go through security and board flights from the terminal at which they checked their baggage in). This may not be the case for transfers where passengers and their baggage arrive at an airport terminal and perhaps leave the airport by another flight departing from a different terminal.

The ABSSAP involves the assignment of BSSs to flights already scheduled. These previously scheduled flights have already been assigned to stands, which are the areas allocated for parking aircraft, and the stand is required from the time of arrival to the time of departure, whereas gates are the areas in a terminal where passengers access the aircraft. In the AGAP, when practitioners refer to the assignment of flights to gates they mean the assignment of the stands associated with these gates, normally located at a pier next to the gate.

3 A model of the problem under study

The problem is composed of N BSSs, and M flights, where flight j requires P_j activities to be completed, each of which must be serviced by a different BSS. The objective here is to find appropriate values for the y_{ijp} Boolean variables, which take a value of 1 if activity p of flight j is assigned to baggage sorting station i, or zero otherwise, with a service starting time s_{jp} and reduction in service time r_{jp} allocated to activity p of flight j. The target service time represents the time in which a BSS is expected to be assigned to a flight. The reduction in service time has a detrimental effect on the robustness of assignments against real-life delays. Therefore, the amount of reduction in the target service time for the assignment of an activity p for flight j is represented by r_{jp}, which is calculated in seconds (as an integer). The described variables are shown in Table 1. A study of different robust-

Table 1 Decision and input variables for this ABSSAP model

Name	Description
y_{ijp}	Specifies the assignment of flights to sorting stations. $y_{ijp} = 1$ if baggage sorting station $i \in [1 \ldots N]$ is allocated to flight $j \in [1 \ldots M]$ for $p \in [1 \ldots P_j]$, and 0 otherwise. If each flight only requires one activity, which means that each flight only requires one BSS, then this variable can be expressed as y_{ij}
r_{jp}	Specifies the necessary reduction in service time for activity $p \in [1 \ldots P_j]$ of flight $j \in [1 \ldots M]$, given the service starting time allocated, s_{jp}
s_{jp}	The service starting time allocated to activity $p \in [1 \ldots P_j]$ of flight $j \in [1 \ldots M]$, and given that a sorting station can only service one flight at a time, s_{jp} can be determined from r_{jp} since $s_{jp} = t_j - r_{jp}$

ness approaches for the ABSSAP has been presented in Ascó (2016).

There are some constraints to be complied with within the ABSSAP:

Assignment Limits each flight must be assigned to at most P_j BSSs, as expressed by Inequality 1.

$$\sum_{i=1}^{N} y_{ijp} \leq P_j \;\; \forall j \in [1 \ldots M] \text{ and } \forall p \in [1 \ldots P_j] \tag{1}$$

Complete Assignment when $P_j > 1$, the activities corresponding to the same flight must either all be assigned or none should be assigned, as expressed by Formula 2.

$$\sum_{i=1}^{N} y_{ijp} = \sum_{i=1}^{N} y_{ij(p+1)} \; \forall j \in [1 \ldots M] \text{ and } \forall p \in [1 \ldots P_j - 1] \tag{2}$$

Reduction in Service BSSs can only be used by one flight at a time, so it may be necessary to reduce the flight service time (usually by reducing the buffer times between flights) in order to assign flights to the same sorting station. The principal objective is usually to maximise the number of assignment of BSSs to flights.

For any pair of different flights where service times overlap, if the overlap in service times is greater than the maximum reduction allowed (B_{lq} for activity q of flight l), then both flight activities cannot be assigned to the same BSS. Thus, Inequality 3 applies to any such pair of flights, j and l ($j \neq l$), where $t_{lq} < e_j \leq e_l$ and $(e_j - t_{lq}) > B_{lq}$.

$$y_{ijp} + y_{ilq} \leq 1 \tag{3}$$

They may otherwise be assigned to the same BSS as long as the service duration of flight l is sufficiently reduced to remove the overlap. Inequality 4 applies to any such pair of flights, j and l ($j \neq l$), and their activities p and q, respectively, where $t_{lq} < e_j \leq e_l$ and $(e_j - t_{lq}) \leq B_{lq}$. One objective is to minimise these service time reductions.

$$r_{lq} \geq (y_{ijp} + y_{ilq} - 1) * (e_j - t_{lq}) \tag{4}$$

Limit of Service Reduction the reduction in service duration may not exceed a limit, as expressed by Inequality 5.

$$0 \leq r_{jp} \leq B_{jp} \quad \forall j \in [1 \ldots M] \text{ and } \forall p \in [1 \ldots P_j] \tag{5}$$

A number of objectives concerning this problem need consideration, and there is a trade-off to be made amongst them. The various objectives considered in this section are:

1. *Maximise Assignment of Baggage Sorting Stations* the first and most important objective is to maximise the number of flights assigned to BSSs, as expressed by Formula 6.

$$f_1 = \max \sum_{i=1}^{N} \sum_{j=1}^{M} \left(\frac{\sum_{p=1}^{P_j} y_{ijp}}{P_j} \right) \tag{6}$$

2. *Robustness* delays on the day of operation may render some assignments infeasible which need to be reassigned. It is therefore desirable to account for potential delays on the day of operation when generating the flight assignments to BSSs at the planning stage, such that the final flight assignments differ little or not at all from the original assignments on the day of operation. The degree to which this is achieved is an indication of the solution robustness, so a solution which requires less in reassignments is said to be more robust than those solutions requiring more reassignments. Robustness is the ability of assignments to resist changes consequence of perturbations by reducing or removing the need to reassign current assignments. One of these approaches is to Minimise Reduction in Service, as expressed by Eq. 7. A study of robustness approaches for the Airport Baggage Sorting Station Problem (ABSSP) is presented in Ascó (2016) and Ascó (2013).

$$f_2 = \min \sum_{j=1}^{M} \sum_{p=1}^{P_j} r_{jp} \tag{7}$$

3. *Minimise Distance* the distance between the BSSs which are assigned to the flights and the flights to which they are assigned should be as short as possible. This objective aims to minimise the inconvenience, work and time involved in getting baggage to the aircraft and could reflect preferences rather than distances. One way to handle this objective would be by expressing it as in Formula 8 where $\sum_{i=1}^{N} \left(y_{ijp} * d_{ij} \right)$ represents the distance between flight j and its allocated BSS

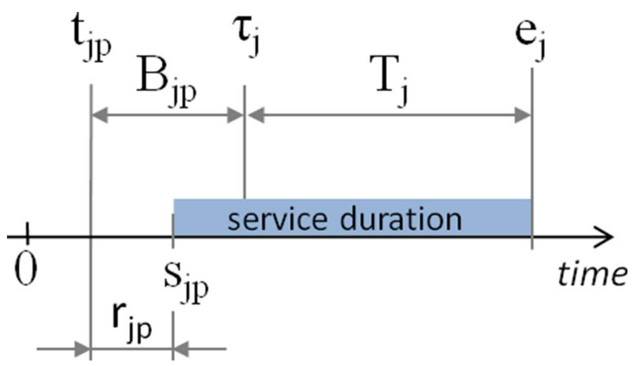

Fig. 3 Representation of the different times

for activity p.

$$f_3 = \min \sum_{j=1}^{M} \sum_{p=1}^{P_j} \left(C_{jp} * \sum_{i=1}^{N} (y_{ijp} * d_{ij}) \right) \quad (8)$$

Other objectives may be considered, such as Consecutive Assignments, Fair Workload, Preferred Piers and Flights to the Same Destination, between others. Some of these other objectives were looked at in Ascó et al. (2013) and Ascó et al. (2011).

A fitness function composed of the weighted sum of the three first objectives presented above was used to guide the search within the algorithm, expressed by Eq. 9 where W_i is the weight for objective i and f_i is the corresponding objective function.

$$f = \sum_{i=1}^{3} W_i * f_i \quad (9)$$

Flights which cannot be assigned to any BSS are assigned to the dummy BSS, an approach widely used in the AGAP, as shown in Tang et al. (2009) , Drexla and Nikulina (2008) and Yan and Huo (2001).

The constants of the model are shown in Table 2, and the relationship between the timing values is illustrated in Fig. 3. A full description of the ABSSAP can be seen in Ascó (2013).

The following two points were defined from the flight density for the day under study and will be observed to be useful later when interpreting the results for the ABSSAP.

The *Lower Maximum Assignment Point (LMAP)* is the number of resources required to service a certain number of activities when the service starting time (s_{jp}) coincides with the target starting service time (t_{jp}), as shown in Fig. 4.

The *Upper Maximum Assignment Point (UMAP)* is the number of resources required to service those activities when the service starting time (s_{jp}) coincides with the base starting service time (τ_j), as shown in Fig. 5.

Table 2 Constants and input values for this ABSSAP model

Name	Description
N	The total number of BSSs under consideration
M	The total number of flights to which sorting stations should be allocated
P_j	The total number of activities to be serviced by baggage service stations for a given flight j, which also equates to the total number of sorting stations required to fully service flight j, $P_j > 0$
T_j	The base service duration for flight j
B_{jp}	The desired buffer time for flight j and activity p ($p \in [1 \dots P_j]$)
e_j	The end service time for flight j
τ_j	The base starting service time for flight j, $\tau_j = e_j - T_j$
t_{jp}	The target starting service time for flight j and activity p, $t_{jp} = \tau_j - B_{jp}$, assuming that the full buffer time is available. Target service duration is the difference between the end service time and the target starting service time, $e_{jp} - t_{jp}$
C_{jp}	A flight specific constant representing the amount of baggage to be processed for flight j and its activity p. This determines the difficulty involved in allocating the flight to a more distant sorting station. For example, this may represent the number of delivery trips required to move the baggage from the sorting station to the aircraft. In the absence of baggage load figures, it was used $C_{jp} = 1$ for all activities and flights
d_{ij}	The distance between baggage sorting station i and flight j
d'_{ik}	The distance between baggage sorting stations i and k

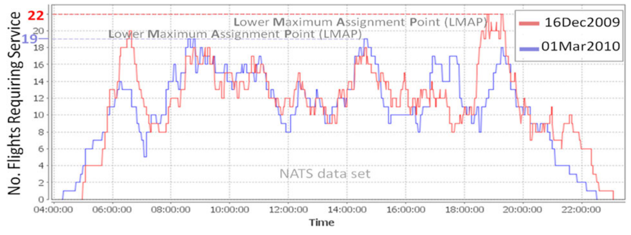

Fig. 4 LMAP for both 16/13/2009 and 01/03/2010

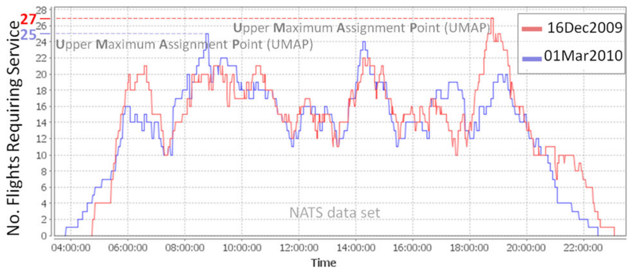

Fig. 5 UMAP for both 16/13/2009 and 01/03/2010

4 Steady-State Evolutionary Algorithm

A Steady-State GA maintains the majority of the population between iterations, only replacing a few individuals at each iteration, a term initially introduced in Syswerda (1989). In the Steady-State Evolutionary Algorithm (SSEA) presented here, Algorithm 1, the next population is obtained by applying the population selection operator, some of which are introduced in Sect. 5-1, to the current population. One of the operators is applied to the individuals selected from the population by the member selector (Sect. 5-2), this last step being called an iteration and being repeated ℓ times, known as a generation. The newly obtained individuals are added to the population so constituting the current population. This is repeated until the termination condition is reached. In contrast to the CGA, parents and offspring typically coexist such that the parents are also considered for the next generation, which theoretically increases the algorithm's ability to retain information for exploitation in subsequent generations. This creates additional selective pressure towards information already contained in the population. However, keeping the parents does not provide the search with new information since it does not sample new genotypes. The approach may incorporate an ageing strategy to ensure that the parents eventually leave the population, thus increasing the chance of offspring contributing to building the next population. Schwefel and Rudolph (1995) incorporated an age by defining a maximum duration of life, so any individual surviving longer than this will be worse than any other which has not reached such limit or has less fitness.

The SSEA is an instance of the Evolutionary Strategies (ESs) which can be described as $(\mu + \lambda)$-ES, $1 \leq \lambda$, where λ may be greater than μ. In the case of the SSEA considered here where the operators used provide only one offspring, when applied, $\lambda = \ell$. A Steady-State GA considering parents in the next generation was presented in Whitley (1989) and Whitley and Kauth (1988), which differs from a CGA in that it uses a serial recombination wherein an offspring replaces the lowest ranking individual in the population rather than the parent, whereas the SSEA may use some or all of the parents in the next generation since the next population in a generation is built by applying the replacement strategy to the current population, which in turn is composed of both the offspring and the parents, so the chance of a parent taking part in the next generation is determined by the replacement strategy used. The SSEA makes use of two selectors, S_p which selects the population which is to take part in the next generation and S_m which selects the member(s) from within an iteration to which the chosen operator is applied. Likewise, Sokolov and Whitley (2005) follows similar steps when generating their GA; the main difference to the SSEA is the use of ℓ, two selection processes and the operators being any combination of operators. The initial population may also be composed

Algorithm 1: SSEA

```
Input: Initial population $P_0$
Input: Number of iterations in a generation $\ell \in \mathbb{Z}^+, \ell > 0$
Input: Operators; $O_j \forall j \in [1 \dots R]$
Input: Replacement strategies, $S_p$
Input: Parent(s) selector, $S_m$
begin
    // Initialisation
    $P \leftarrow P_0$; // set inital population
    // Execution of generations
    repeat
        $P \leftarrow S_p(P)$; // apply replacement strategy
            to get the new population
        $P_t \leftarrow \emptyset$; // empty population of children
        $i = 1$; // initialise the iterations
        // Execution of iterations
        repeat
            Select an operator, $O_k$;
            $Q \leftarrow S_m(P, O_k)$; // select parents
            $Q \leftarrow O_k(Q)$; // generate children
                solutions by applying operator
            $P_t \leftarrow P_t \cup Q$; // add children solutions
                to the population of children
            $i = i + 1$; // increment iteration
        until $i \leq \ell$ or Termination Condition;
        $P \leftarrow P \cup P_t$; // merge parents with
            children solutions
    until Termination Condition;
    return $P$;
end
```

of fewer solutions than the preferred population size. This size should eventually be reached as the new solutions generated are merged with the parent solutions, and then, the replacement strategy is applied.

For $\ell = \mu$ (the population size), the SSEA algorithm is closer to a CGA, but still differs from the CGA in that:

1. The new population to which the replacement strategy is applied is of size $\mu + \lambda$, whereas for the CGA it is λ. Thus, not only do parents and offspring coexist in the new population, but also those previous solutions which may not have been selected for the generation of offspring in the current generation.
2. A generation is composed of ℓ iterations in which parents are selected and operators applied to generate the offspring, which together with the previous population will compose the current population. ℓ does not need to be fixed, and it can be changed as the search progresses, thus providing an additional mechanism to control the sampling.
3. Whereas in the CGA reproduction produces two offspring, in the SSEA the reproduction may produce either one or many offspring.

4. In the CGA up to two operators may be applied, namely crossover and mutation. The SSEA does not put any restriction on the operator, so operators may be applied one per iteration or a set of operators in an iteration, as described in the following sections. An operator may be defined which sequentially applies a set of sub-operators to the offspring of the previous operator, based on some criterion, such as the probability of a sub-operator being selected. An example of this is where two operators are used one with a probability of 1, so it is always used, and a second operator a probability of 0.1 being used. The first offspring is always obtained by applying the first operator to the parents in the population, given its probability of 1. This offspring may be further modified by the second operator in order to obtain the final offspring; otherwise, where the second operator is not applied, the first offspring becomes the final one. If both probabilities are lower than 1, there is a chance of the parent also becoming the final offspring.
5. Each operator has a probability associated with it which represents the chance of being selected, where the overall probability of selecting any of the operators totals 1. In this SSEA, any of the operators may be selected at each iteration based on their probabilities.

5 Selectors

The selector methods are responsible for selecting solutions within a population of solutions. Two types of selector are used throughout this paper which are:

1. *Replacement Strategies* (S_p) The replacement strategies generate the new population from the parents and offspring which is used in the following generation. The replacement strategies are used in both CGA and SSEA. They distribute the chance of individuals taking part in the next generation. Normally, the fitter the solution, the more chance it has of being selected for participation in the following generation. A comprehensive analysis of selection schemes used in EAs can be found in Blickle and Thiele (1996).
2. *Parent Selectors* (S_m) The member selectors distribute the chance of a given solution within the population taking part in generating new offspring within a generation. Normally, the fitter the solution, the more chance there is of being selected to produce new offspring.

An increase in diversity certainly corresponds to broadening exploration of the search space, and finding an adjustable balance between exploration and exploitation is the key, Levinthal and March (1993) and March (1991). Exploration and exploitation should not be constrained to specific parts of the process, such as only in the early stages of the search, but also be taken into account throughout all the evolutionary processes based on the characteristics at each stage.

The selection of solutions for participation in a population is one of the mechanisms for managing diversity, which together with the operators helps to improve the direction of the search within the domain of solutions into the regions containing solutions with a higher potential.

Some of the terms used are defined below which are based in Baker (1987) and Blickle and Thiele (1996).

– *Selective pressure* is the probability of selecting the best individual compared to the average probability of selection of all the individuals.
– *Bias* is the absolute difference between an individual's normalised fitness and its expected probability of reproduction.
– *Spread* is the range of possible values for the number of offspring of an individual.

There follows an overview of the new approaches proposed.

5.1 Stochastic Universal Modified Sampling

The Stochastic Universal Sampling (SUS) may not be appropriate when the order of magnitude of the fitness under study is greater than the difference in the fitness values amongst individuals; such are the cases studied in this paper. So Stochastic Universal Modified Sampling (SUMS) is defined in such a way as to provide a greater selection pressure, as shown in Algorithm 2. SUMS provides more selection pressure than SUS and some bias. A characteristic of the SUMS is that offsetting of all of the fitness by a constant does not affect those sections of the roulette wheel occupied by each solution as this is not the case for the SUS.

In both versions, a single spin of the roulette wheel is made which provides both a starting point and the first individual. The following selections are made by advancing the point in equal step sizes and selecting the individual occupying the section upon which the point fell: the process is repeated until all the required individuals have been selected. Some individuals may not be selected where their occupied section is sufficiently small, depending on the starting point.

Both versions of sampling ensure that the observed selection frequencies of each individual are in line with the expected frequencies. So if there is an individual occupying 6.5% of the wheel and it is necessary to select 100 individuals, it is expected, on average, that that individual will be selected between six and seven times. Whereas both SUS and SUMS guarantees this, roulette wheel selection does not make such a guarantee.

Algorithm 2: Stochastic Universal Modified Sampling

```
Input: Population $P$ of size $\lambda$
Input: Desired population size of $\mu$, $0 < \mu < \lambda$
begin
    // Calculate the two lowest fitness
    $F_{min} = \infty$;
    $F_{min-1} = \infty$;
    for $i = 1 \rightarrow \lambda$ do
        if $F_{min} > f_i$ then
            $F_{min-1} = F_{min}$;
            $F_{min} = f_i$;
        end
        else if $F_{min} > f_i$ then
            $F_{min-1} = f_i$;
        end
    end
    $F = F_{min} - (F_{min-1} - F_{min})$;

    // Assign a section to each solution
    $p_0 = 0$;
    for $i = 1 \rightarrow \lambda$ do
        $p_i = \frac{\sum_{j=1}^{i}(f_j - F)}{\sum_{j=1}^{\lambda}(f_j - F)}$;
    end

    // Initialise
    $P' \leftarrow \emptyset$; // empty next population
    $r_0 = rnd\left[0, \frac{1}{\mu}\right)$; // identify first point
    $i = 1$; // set to first solution in P

    // Select members from the population
       based on their roulette wheel section
    for $j = 1 \rightarrow \mu$ do
        $r = \frac{(j-1)}{\mu} + r_0$;
        for $i \rightarrow \lambda$ do
            if $p_i > r$ then
                $P' \leftarrow i$; // add selected solution to
                   next population
                break;
            end
        end
    end

    return $P'$;
end
```

5.2 Index selector (ISxy)

This new selector makes sure that no more than a fixed maximum number of fitness duplicates are selected for the next population. This selector requires an integer which corresponds to the maximum number of solutions with the same fitness to keep (x, number of solutions) and a base selector (y, the base selector), one of the selectors presented above, e.g. the Index Selector with the Elitist Selector and a group size of 1 would be represented as $IS1ES$.

The Index Selector is only useful as a replacement strategy, given that as a parent selector it merely selects a very reduced number of solutions.

5.3 Range index selector (RISxyz)

Empirical results show that when the previous selector ISxy was applied to the ABSSAP, different groups with small differences were generated, which also represented a reduction in diversity, and which diversity may be increased further by changing the ISxy from a unique fitness in each group to a range of fitness per group. This requires a knowledge of group size (x, the maximum number of solutions to be kept within a range), a base selector (y, the base selector) and an indication of the fitness range (z), e.g. the Range Index Selector with Elitist Selector (y = ES), a group size of 1 ($x = 1$) and fitness range of 50 ($z = 50$) which may be represented as $RIS1ES50$. For $RIS1ES50$ and a maximisation problem, if the group having a fitness range from 1000 to 1050 already contains a solution with a fitness of 1000, and a new solution is to be added to the population with a fitness of 1010, then the solution of a 1000 is removed and the new solution is introduced into the group in its place, given that $x = 1$. The selection within a group uses a greedy approach.

Many of the selection approaches presented are not suitable where only one individual (solution) is required, as is the case for the Index Selection (ISxy) and Range Index Selector (RISxyz), given that in those cases they are equivalent to the underlying selection approach, e.g. the Index Selection with Elitist Selection (ISxES) is the same as the Elitist Selection (ES). Such is the case for the mutation operators (Sect. 6) where the Parent Selectors have to select only one parent solution. Similarly, some of the classic selection methods such as SUS, Roulette Wheel Member Selection (RWMS) and Tournament Member Selection (TMS) are equivalent when just one parent solution has to be selected.

6 Operators

Two main groups of operators are reviewed in the following sections: Mutation and Crossover. Both of these are described below.

6.1 Mutation

The operators introduced here are local search (guided mutation) operators which generate feasible solutions.

All flights which have not been assigned to a sorting station are assigned to the 'dummy' sorting station. Some operators can switch flights between the real and dummy sorting stations.

When a sorting station is to be selected, the roulette wheel selection method is used whereby every sorting station has the same probability of being selected.

When a time has to be determined (for instance for the start or end of a time range), a uniform random variable is used, so that any time within the time range of the flights under consideration has an equal probability of being chosen.

6.1.1 Dummy Single Exchange Mutation Operator (DSEMO)

The DSEMO is equivalent to the 'Apron Exchange Move' used by Ding et al. (2004, 2005). A solution is selected from the population by the member selector (S_{m}); then, a new solution is built by moving a flight from the 'dummy' sorting station in this solution to a randomly selected sorting station, thus potentially moving another flight back onto the 'dummy' sorting station where it can no longer be fitted in.

This operator may increase the number of assignments where the operation does not move a flight back onto the 'dummy' sorting station.

It is necessary that some flights be unassigned in the parent solution. So when full assignment has been attained for the given number of BSSs, this operator clearly will not provide a new solution.

6.1.2 Dummy Single Move Mutation Operator (DSMMO)

In the DSMMO, a random unallocated flight and initial target sorting station are chosen, and an attempt is made to assign the flight to the selected sorting station. If the assignment cannot be achieved, then the next sorting station is selected and the process is repeated until the flight is assigned, or no more sorting stations are available, in which case the flight is returned to the 'dummy' sorting station. When maximum assignments have been attained for the given number of sorting stations, this operator obviously will not provide a new solution.

6.1.3 Multi-Exchange Mutation Operators

A set of sorting stations is randomly selected from these operators within a random time period, t_{rs} to t_{re}. All assignments where the base service durations are entirely within the time period are then moved to the next sorting station in the set, as shown in Fig. 6, provided they fit. This operation is repeated from one sorting station in the set to the next, until they have all been covered. Flights which cannot be moved are added to the set of flights which will be considered for assignment at the end, potentially reducing the number of flights which would otherwise not be assigned. These operators generalise the 'Interval Exchange Move' which was presented by Ding et al. (2005), and cannot increase the number of assignments.

Three variants have been developed:

1. Multi-Exchange between a Fixed Number of Resources (MEFNRn): The number of sorting stations between which flights are exchanged is fixed at n, where $2 \leq n \leq N$.

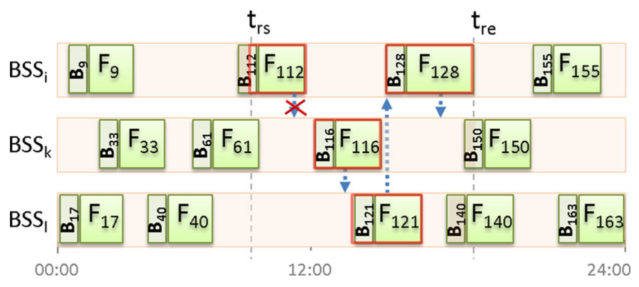

Fig. 6 Example of multi-exchange between three BSSs

2. Multi-Exchange between a Random Number of Resources (MERNRn): The number of sorting stations between which flights are exchanged is randomly chosen each time, between 2 and n, where $2 < n \leq N$.
3. Multi-Exchange between a Range Random Number of Resources (MERRNRxy): The number of sorting stations between which flights are exchanged is randomly chosen each time, between x and y, where $2 \leq x < y \leq N$.

6.1.4 Multi-Exchange By Pier Mutation Operators

These operators are a specialised case of the Multi-Exchange Mutation Operators, where the sorting station selection element ensures that no two consecutive sorting stations in the set are on the same pier. The idea is to improve the distance objective by encouraging the movement of assignments between piers.

Once again, this operator cannot increase the number of assignments. As for the Multi-Exchange Mutation Operators, three variants have been created:

1. Multi-Exchange By Pier between a Fixed Number of Resources (MEBPFNRn): The number of sorting stations to exchange flights between is fixed at n, where $2 \leq n \leq N$.
2. Multi-Exchange By Pier between a Random Number of Resources (MEBPRNRn): The number of sorting stations between which the flights are exchanged is randomly chosen each time, between 2 and n, where $2 < n \leq N$.
3. Multi-Exchange By Pier between a Range Random Number of Resources (MEBPRRNRxy): The number of sorting stations between which the flights are exchanged is randomly chosen each time, between x and y, where $2 \leq x < y \leq N$.

6.1.5 Range Multi-Exchange Mutation Operators

These are the same as the Multi-Exchange Mutation Operators; however, they add an additional feasibility recovery step when flights cannot be moved. Flights which cannot be

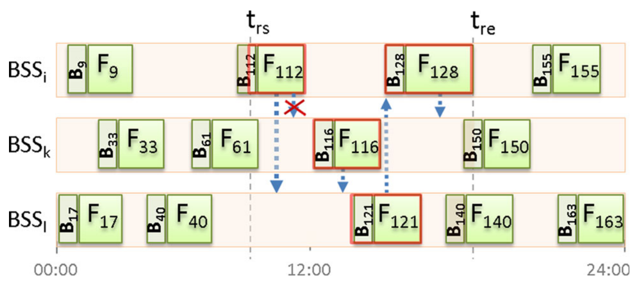

Fig. 7 Example of range multi-exchange between three BSSs

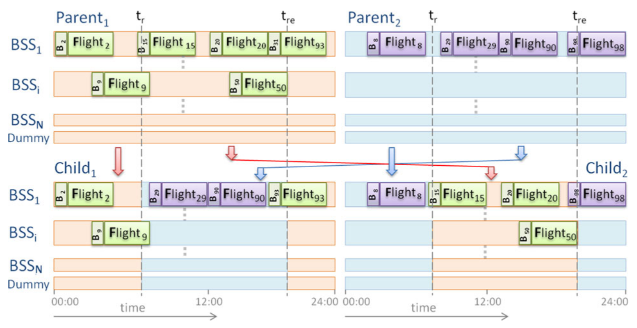

Fig. 8 2-Point crossover

moved are added to the set of flights which will be considered for assignment to the next sorting station, potentially reducing the number of flights which would otherwise not be assigned at the end. Finally, flights which have still not been moved are again considered for assignment to the other sorting stations in the set, except the last one, once again potentially reducing the number of flights which otherwise would not be assigned, in the same way as the Multi-Exchange Mutation Operators, as shown in Fig. 7.

Once again, this operator cannot increase the number of assignments. Three variants have been developed:

1. Range Multi-Exchange between Fixed Number of Resources (RMEFNRn): The number of sorting stations between which to exchange flights is fixed at n, where $2 \leq n \leq N$.
2. Range Multi-Exchange between Random Number of Resources (RMERNRn): The number of sorting stations between which to exchange flights is randomly chosen each time, between 2 and n, where $2 < n \leq N$.
3. Range Multi-Exchange between Range Random Number of Resources (RMERRNRxy): The number of sorting stations between which to exchange flights is randomly chosen each time, between x and y, where $2 \leq x < y \leq N$.

6.1.6 Range Multi-Exchange By Pier Mutation Operators

These are a specialised version of the Range Multi-Exchange Mutation Operators, which ensure that consecutive sorting stations in the set are not on the same pier, to encourage the movement of flights between piers, so potentially improving the distance objective. These operators cannot increase the number of assignments. As for the Multi-Exchange Mutation Operators, three variants have been created: Range Multi-Exchange By Pier between Fixed Number of Resources (RMEBPFNRn) and with Random Number of Resources Range Multi-Exchange By Pier between Random Number of Resources (RMEBPRNRn) and Range Multi-Exchange By Pier between Range Random Number of Resources (RMEBPRNRxy).

The Multi-Exchange Mutation Operators may also be extended by using multiple points in time instead of two points in time (a time range). However, this will also increase the complexity and time required to execute the operations and equates to several executions of the current implementation and was not therefore investigated.

6.2 Crossover

The crossover operators involve the generation of new solutions from multiple parents. Each parent will be chosen using the Parent Selectors (S_m), and multiple child solutions may be generated in each case.

6.2.1 2-Point crossover

In the 2-point crossover (C2P), two points in time are randomly selected within the time range of the flights, to generate a time window. All flight assignments which lie within this time period, for all of the sorting stations in each solution, are exchanged between the parent solutions, as shown in Fig. 8. The flight timings are identical across all solutions, except that the flights in the exchanged region may overlap flights which are not exchanged in the case of some sorting stations. Such overlapping flights in the exchange region are reassigned to other sorting stations where possible; otherwise, they are assigned to the dummy sorting station (i.e. are unassigned).

Whereas in the classic crossover a chromosome is divided into three sections, here the chromosome is divided into $3 * N$ sections which correspond to three sections per sorting station.

6.2.2 1-Point crossover

The 1-point crossover (C1P) is a specific case of the above 2-point crossover, where the window extends to the end time of the solution, as shown in the left in Fig. 9.

In the presented representation, 1-point crossover is a special case of 2-point crossover ($n = 2$, number of points),

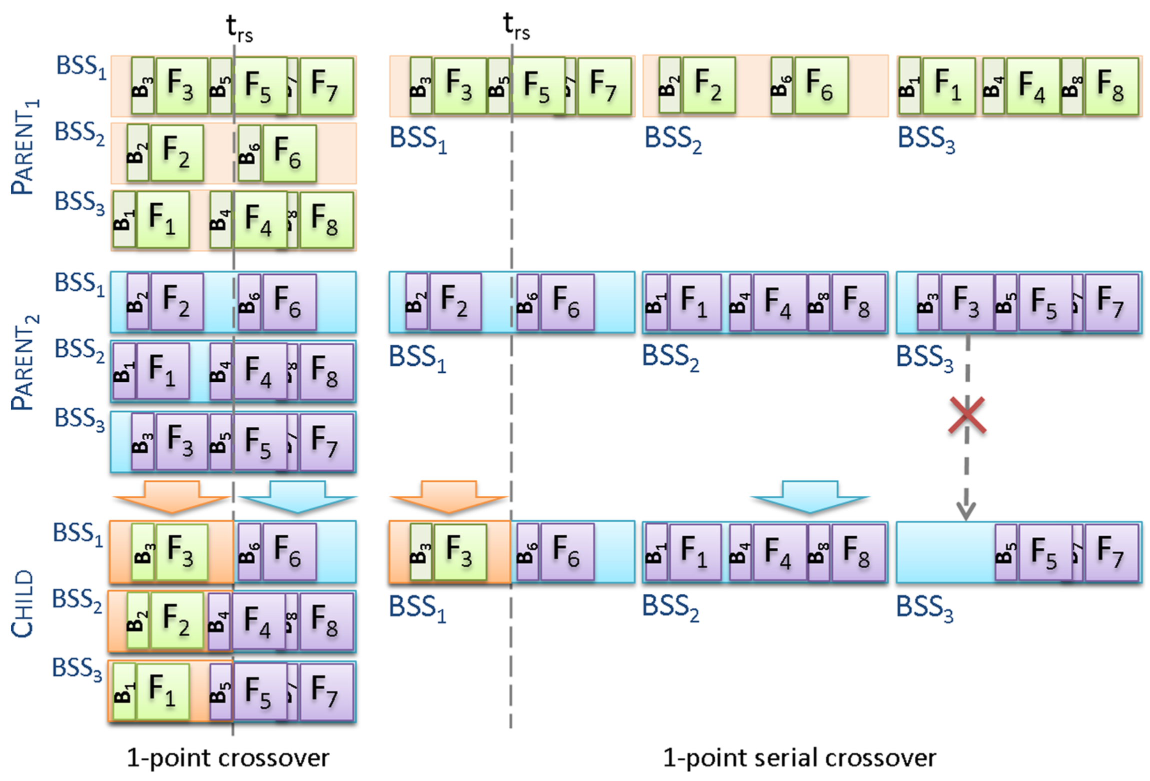

Fig. 9 Example of 1-point crossover and 1-point serial crossover

where the second point corresponds to the end of the chromosome.

6.2.3 n-point crossover

The n-point crossover (CnP) may use $n + 1$ solutions from the population, where n refers to the number of cuts. The full time range is divided into $n + 1$ sections, and multiple new solutions are obtained by merging the consecutive sections between the different parents. This recombination may leave some flights unassigned, which may be assigned directly to the dummy sorting station (fictitious sorting station) or an attempt could be made to repair the solution by assigning it to any available sorting station. An extension to 2-point crossover is the n-point crossover which divides the chromosome into $(n + 1) * N$ which equates to $n + 1$ sections per sorting station.

Using n-point crossover with $n+1$ parents can provide up to $(n+1)!$ children. Eiben et al. (1994, 1995), Tsutsui and Jain (1998) and Eiben (2003) studied the effect of using multiple parents and multiple crossover points and observed that the increase in the success rate is not merely a consequence of using multiple crossover points, leading to the conclusion that using more parents does increase GA performance.

6.2.4 1-Point serial crossover

The 1-point serial crossover (SC1P) is a different implementation of a crossover operator and may be simpler to understand by representing the problem as a continuous list of BSSs where the crossover cut(s) is in this continuous list, instead of within each BSS as seen in the previously presented crossover operators. The 1-point serial crossover operator is illustrated on the right in Fig. 9 for the ABSSAP. When the cut(s) has to be determined, a comparison of both parents is made to find the first and last differences in their assignments within the representation, which may be used to restrict the selection of the cut(s). This implementation of a crossover operator is closer to that which is commonly presented in the literature as a 1-point crossover operator, and it is different to that previously introduced in this section.

Figure 9 shows a simple example where the same parent solutions are used in a 1-point crossover and a 1-point serial crossover side by side. When considering two parents with full assignment, the cut in time (t_{rs}) in the 1-point crossover (C1P) breaks the assigned flights into two groups, each of which contains the same flights for both parents, whereas this is not the case for 1-point serial crossover (SC1P), as shown in Fig. 9, where flight ‘3’ is on a different side of the cut in the parents. This means that in the case of SC1P it is necessary to check the assignments after the cut (t_{rs}) from the second parent to make sure that they have not already been assigned to the first side (from the first parent). Flight '3' is already assigned to the offspring of the first parent and therefore cannot be assigned again to those from the second parent, as shown in Fig. 9. So 1-point crossover is simpler to implement than 1-point serial crossover.

Furthermore, this implementation could easily be extended to n points.

Holland (1975) argued that, based on the schema theorem to minimise schema disruption, 2-point serial crossover is better than 1-point serial crossover. Although our results show that in some instances 1-point serial crossover provides better solutions than 2-point serial crossover, in general 2-point serial crossover performs best overall. Nevertheless, the schemata theorem is based on a binary representation of the chromosome and binary operators, which differ from the representation and operators presented here, so its application is of limited interest.

6.3 Combination of operators

Based on how the operator is selected, the types which are of interest are described in the following subsections. It is noted that the operators could be used in complex ways by combining these different types with different parameters.

6.3.1 Probability Single Multi-Operator

The Probability Single Multi-Operator (PSMO) is composed of several sub-operators (which are described in Sect. 6), each one of which has a specified probability of being used for the creation of new population members, as shown in Algorithm 3. The combined probabilities across all operators must add up to 1.

As an example, consider a PSMO operator which uses the operators C1P (with a 0.1 probability of being selected) and Multi-Exchange between a Fixed Number of 3 Resources (MEFNR3) (with a 0.90 probability of being selected), which may be represented as PSMO(C1P:10+MEFRN3:90). Given that the total probability must amount to 1, it is not necessary to specify the probability for the last sub-operator, so the representation may also be PSMO(C1P:10+MEFRN3).

The PSMO operator is the one used as base operator in the SSEA experiments presented in Sect. 8.

Algorithm 3: Probability Single Multi-Operator.

```
Input: Member Selector $S_m$
Input: Population of solutions $P$
Input: Operators; $O_k \; \forall \, k \in [1 \dots R]$
Input: Probability for operators $p_k$, $0 < p_k \leq 1 \forall k \in [1 \dots R]$
       and $\sum_{k=1}^{R} p_k = 1$
begin
  // Initialise
  $P_0 \leftarrow \emptyset$; // empty list of children
  $r = rnd[0 \dots 1)$;
  $k = 1$; // initialise sub-operator index to
     first operation
  $p = p_1$;

  // Select operator
  while $k < R$ and $r > p$ do
     $k = k + 1$; // next operator
     $p = p + p_k$;
  end
  $Q \leftarrow S_m(P, O_k)$; // get parent solutions for
     operator $O_k$
  $P_0 \leftarrow O_k(Q)$; // build children by applying
     operator to parents

  return $P_0$; // return the obtained children
end
```

Algorithm 4: Sequential Operator.

```
Input: Member Selector $S_m$
Input: Population of solutions $P$
Input: Operators; $O_k \; \forall \, k \in [1 \dots R]$
Input: Probability for each operator $p_k$,
       $0 < p_k \leq 1 \; \forall \, k \in [1 \dots R]$
begin
  // Initialise
  $P_0 \leftarrow S_m(P, O)$; // select parents based on
     operators

  // Build children
  for $k = 1 \rightarrow R$ do
     $r = rnd[0 \dots 1)$;
     if $r < p_k$ then
        $Q \leftarrow P_0$; // previous children as
           parent solutions
        $P_0 \leftarrow O_k(Q)$; // applying operator to
           the parent solutions
     end
     $i \leftarrow k + 1$; // next sub-operator
  end

  return $P_0$; // return the obtained children
end
```

6.3.2 Sequential operator

Considering the way the CGA operates, where a crossover operator may be applied to the parents with a high probability and its children may be further modified by applying a mutation operator, the operators may be extended by defining a new operator composed of multiple sub-operators, which are applied sequentially with a given probability ($0 < p \leq 1$), as shown in Algorithm 4. This new operator is called the Sequential Operator (SO) herein.

As an example, consider the operators C1P with a selection probability of 1 and the MEFNR3 with a probability of selection of 0.01, which may be represented as SO(C1P:100,MEFNR3:1), where a 1-point crossover is always applied to generate the intermediate children. For these, there is a small probability of 0.01 for application of

Table 3 Default parameter values

Parameter	Value	Comments
Tournament size	2	Tournament selection
Trails/runs	30	Number of runs per experiment
Significance level	0.05	Mann–Whitney U tests were carried out to ascertain the statistical significance
Fitness weights	$W_1 = 90 \quad W_2 = -0.008 \; W_3 = -1$	Also used in Ascó et al. (2012), and the weights calculation can be seen in Ascó (2013)

the MEFNR3 operator in order to generate the final children solutions.

7 General experiments information

A summary of some of the typical values for the different parameters used in the following experiments is shown in Table 3.

The data sets used relate to those provided by NATS Ltd. both for 16 December 2009 (H1T091216) and for 1 March 2010 (H1T100301).

The initial solutions were obtained by running the constructive algorithms presented in Ascó et al. (2010, 2011).

Unless it is mentioned, the parameters presented here refer to all the following experiments for the ABSSAP.

8 Results

The algorithms described are applied to the ABSSAP for different number of BSSs and stands, and their results are compared and analysed in this section for both the data sets obtained from British Airports Authority (BAA)'s website and those provided by NATS Ltd. for Heathrow Airport London. A fitness function composed of the weighted sum of the different objectives was used to guide the search within the algorithms.

Normality tests were run to identify whether the data could be said to follow a normal distribution, which is a requirement for use of the t-test; otherwise, the Mann–Whitney U test is preferable. Razali and Wah (2011) compared some normality tests and concluded that Shapiro–Wilk is the most powerful normality test. Thus, the Shapiro–Wilk normality test was run for some of the data to ascertain whether the data could be said to be normal, but the data could not be said to follow a normal distribution, the results of which can be seen at Ascó et al. (2013). So Mann–Whitney U tests were carried out to ascertain the statistical significance in the following experiments.

The initial results from experiments executed for BAA's website data sets for Heathrow Airport London show that the SSEA presented here provides better solutions than those obtained by CPLEX and Gurobi for the running times considered. These experiments also highlighted the need to have access to a large quantity of Random Access Memory (RAM) given how memory hungry both commercial solvers CPLEX and Gurobi are, making it necessary to run them on a 64bit machine to be able to use more RAM. An initial run with a duration of 1 h was executed, followed by another of 24 h to identify whether the exact method could find the optimum and compare the fittest solution obtained with those obtained by the SSEA. Also the best upper bound obtained from each run was used to help to get an idea of the of the solutions quality obtained from the different algorithms used in the following sections. All the Gurobi parameters used were the default ones with the exception of the time, which was limited to 1 h and 24 h in the two initial runs, and the parameter values used for CPLEX are presented in Table 4. Multiple runs were executed to enable the SSEA to take account of the random characteristics of the algorithm with a PSMO composed of 0.2 MEFNR3, 0.2 RMEFNR2, 0.15 C1P and 0.45 DSEMO (only one of the sub-operators will be used at each iteration) with an ES replacement strategy, and the results are shown in Fig. 10.

Table 4 CPLEX none default parameter values used for results in Figs. 10 and 12

Parameter	Value	Comments
NodeFileInd	3	Node file on disc and compressed
WorkMem	128	Memory in MB
NodeSel	1	Best-bound search
VarSel	3	Strong branching
TiLim	3600 and 86400	Time in seconds to end the run

The SSEA quickly improves upon the initial solutions used, reaching solutions fitter than those obtained by Gurobi. Further initial experiments were conducted between the SSEA, CGA and TS with the parameter values as shown in Table 5. The results for these experiments, which are presented in Fig. 11, also show that SSEA performs better than the other Metaheuristics considered.

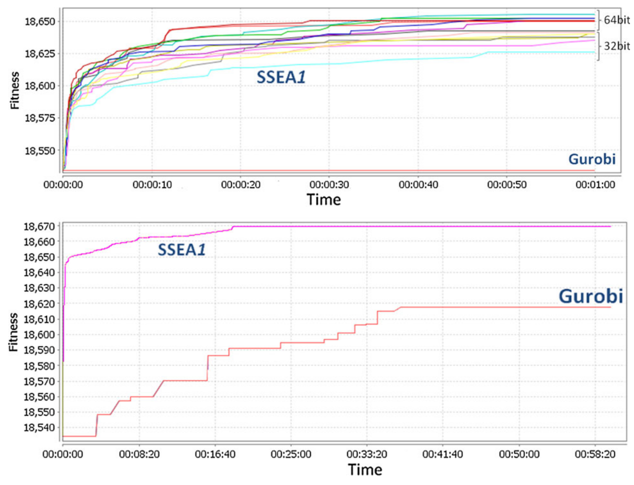

Fig. 10 Progress for a 3-pier topology, 48 stands, 78 BSSs and 219 flights (H1T091216)

Table 5 Parameter values used with 30 runs per experiment

Algorithm	Parameters	
	Name	Value
SSEA	Population size	10 and 30
	Tournament size	5
	Replacement strategy	ES
	Operator	MEFNR3
CGA	Population size	10 and 30
	Replacement strategy	ES
	Operator	0.99 C1P and 0.1 MEFNR3
TS	Walk size	10
	Tabu list size	30
	Operator	MEFNR3

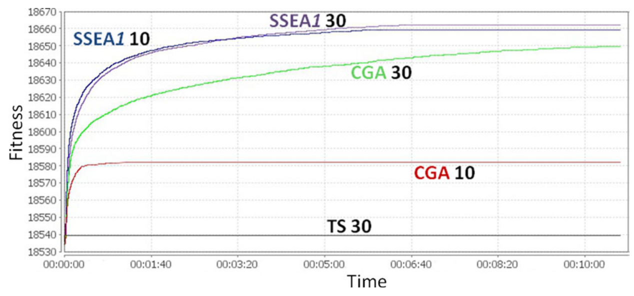

Fig. 11 Progress for a 3-pier topology, 48 stands, 78 BSSs for 219 flights (H1T091216) and different heuristics

In general, the results obtained show improvements in fitness, as shown in Fig. 12. Better results were obtained when other Replacement Strategies were used, which are presented in the following sections.

These results show the potential of the SSEA for obtaining good solutions even on short runs. It may also be noted that the problem becomes simpler as the number of BSSs increases, especially for the number of BSSs near or bigger than the UMAP, where there are sufficient BSSs to service all the flights while keeping the buffer time intact.

In the next sections, the experiments and their results are presented which were obtained when studying the different parameters part of the SSEA.

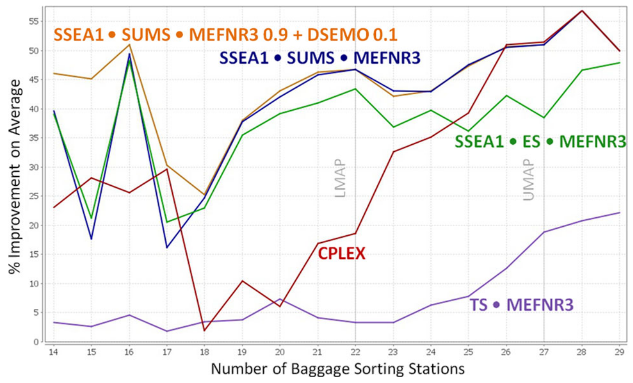

Fig. 12 Average fitness for a 4-pier topology, 46 stands for 194 flights (H1T091216) and different heuristics

8.1 Initial solutions

Experiments were initially conducted to evaluate the influence of the initial population of solutions in reaching better solutions when using good solutions as initial population. The latter have been obtained by applying the constructive algorithms presented in Ascó et al. (2010, 2011), to a data set of 219 flights. The operator used is a PSMO composed of the following sub-operators, each with its own probability of being used; 0.2 for RMEFNR*n*, 0.2 for Dual Exchange Mutation Operator (DEMO), 0.15 for 1-point crossover and 0.45 for DSEMO, for a population size of ten solutions for a population based algorithms and 78 BSSs (lower than the LMAP). Given that for 78 BSSs full assignment is not possible, use of the DSEMO should help to reach other areas of the search space, thereby improving the solutions obtained. The solutions which do not have maximum assignment may further increase the number of assignments by applying the DSEMO.

Maximum assignment is achieved where no buffer time is considered, and no restriction is applied as to where the flights may be assigned when ordering the flights by departure time: this is used to generate some of the constructive solutions. The progress of the search is used here for the different initial solutions being considered, in order to illustrate their contribution in reaching better solutions, as shown in Fig. 13. This provides a view of the Steady-State Evolutionary Algorithm with $\ell = 1$ (SSEA1) behaviour and shows that the algorithm managed to improve on the already good solutions provided as initial solutions, but not as much

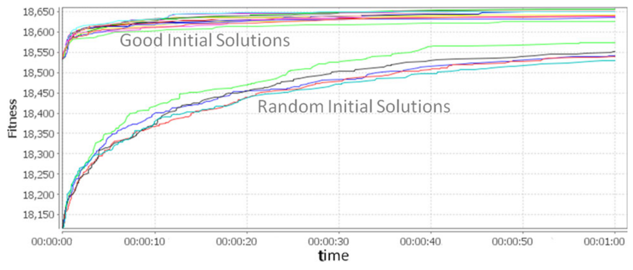

Fig. 13 Progress in fitness of solutions when run with and without a initial random population for SSEA1, a 3-pier topology, 78 BSSs, 48 stands and 219 flights (H1T091216)

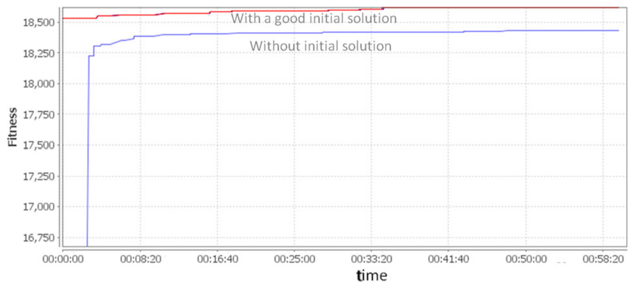

Fig. 14 Progress in fitness of solutions when run with and without initial random population for Gurobi, a 3-pier topology, 78 BSSs, 48 stands and 219 flights for 1 h

as when the initial solutions are of lower fitness. This is as expected given that there is more leeway to improve on the solutions, but the final fitness of the best solutions is still less than those obtained when good solutions are used. Furthermore, the solver Gurobi was run for 1 h, as shown in Fig. 14, when no initial solution was provided and when an initial constructive solution (the best of those used for the SSEA1) was used, which showed Gurobi took over 2 min to find a feasible initial solution, when no initial solution is provided. Then quickly improved on this, but still does not manage to reach a fitness such as those reached when a good constructive initial solution is provided, as is the case with the SSEA1, but at a lower rate. The final solution fitness in both figures shows that SSEA1 provides fitter (better) solutions than those provided by Gurobi, with SSEA1 also improving on Gurobi when no good initial solutions were used.

In summation, the benefits of using good initial solutions in the SSEA are more apparent at short running times, as the differences between fitness decrease as the running time increases, but fitter overall solutions are found when the algorithm uses fit good initial solutions. This was also noted when using commercial optimisation applications such as Gurobi and CPLEX.

The mutation operators considered here, with the exception of DSEMO, cannot increase the number of assignments; therefore, solutions which do not have maximum assignment restrict the search space and waste iterations which could otherwise be used to widen the search of the space of solutions potentially improving on those solutions already found. This can be particularly detrimental if none of the solutions provided are sufficiently fit, i.e. solutions with at least one unassigned flight which is assigned in the optimal solution, as such flights cannot be assigned by these operators. Therefore, when the initial solutions do not have maximum flight assignment for the given number of BSSs, the search is restricted to flights already assigned which means low fitness. In these cases, the use of another operator which can increase the number of assignments such as the DSEMO should be used, at least until one or many of the solutions in the population reach maximum flight assignment.

Table 6 shows the statistical fitness significance of the best solution obtained by the SSEA1 with a population size of 30 and a single operator MEFNR3, when using an initial population composed of good solutions obtained from applying the constructive algorithms studied in Ascó et al. (2010, 2011). It is compared with those solutions obtained when the initial population is composed of the 30 fittest solutions from 200 randomly generated solutions (random constructive algorithm) for the data set. The empirical results show that the SSEA with good initial solutions provides, in most of the instances considered here, a superior final best solution (statistical fitness significance < 0.005) than when the initial population is composed of random solutions.

The algorithms in the study in the following sections use the initial solutions obtained by applying the constructive algorithms which were used in this section and introduced in Ascó et al. (2010, 2011).

8.2 Population size

The effect of the population size (μ) on the results of several of the operators presented in 6 (Operators) was explored. The parameters used in the experiments are:

1. The data sets used relates to those provided by NATS Ltd. both for 16 December 2009 (H1T091216) and for 1 March 2010 (H1T100301), with both 3-pier and 4-pier topologies, for Heathrow Airport London.
2. Number of BSSs of $N \in [13 \dots 29]$.
3. The operators used are: C1P, C2P, DSEMO, Multi-Exchange By Pier between a Fixed Number of 3 Resources (MEBPFNR3), MEFNR3 and RMEFNR2. The number of resources (BSSs) considered for the mutation operators used was determined by a comparison of the initial results obtained from runs with a population size of 30 for each of the mutation operators.
4. The number of iterations per generation ℓ was initially set to 1.
5. The replacement strategies used are: ES, SUMS, Index Selection with Elitist Selection and a group size of 1

Table 6 Statistical fitness significance for a significance level of 0.05, SSEA1 with fit initial solutions and initial random solutions for the data sets provided by NATS Ltd

Data set	3-Pier								
	13	14	15	16	17	18	19	20	21
194 flights	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
H1T091216	22	23	24	25	26	27	28	29	
	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
	13	14	15	16	17	18	19	20	21
163 flights	0.0000	0.0686	0.7746	0.0000	0.0000	0.0000	0.1127	0.0000	0.8741
H1T100301	22	23	24	25	26	27	28	29	
	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
Data set	4-Pier								
	13	14	15	16	17	18	19	20	21
194 flights	0.0000	0.0000	0.0000	0.0000	0.0000	0.1275	0.0000	0.0000	0.0000
H1T091216	22	23	24	25	26	27	28	29	
	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	
	13	14	15	16	17	18	19	20	21
163 flights	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	0.0000	0.0000
H1T100301	22	23	24	25	26	27	28	29	
	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	

(IS1ES) and Index Selection with Stochastic Universal Modified Sampling and group size of 1 (IS1SUMS).

6. Population sizes of $\mu \in \{1, 5, 10, 15, 30, 50, 100, 200, 500, 800, 1000, 2000\}$ were considered. The algorithm was initially run for population sizes of 15, 30, 50, 100, 200, 500 and 1000. In some instances, the best values appeared at the end of the ranges, which encouraged extending the range of population sizes studied according to the best population size for each of the operators types.
   Regarding the Multi-Exchange Operators, only extra population sizes of 1, 5, 10 were studied, given that these operators are guided mutation operators based on chance and provided better results for the lowest population sizes initially considered. Nevertheless, given the poor results obtained when using the TS, as shown in Fig. 11, it was anticipated that the size of the population should be higher than 1.
   In the case of the DSEMO, the results indicated that a high population size was preferred, such that other appropriate population sizes were then considered. The population sizes of 500 and 1000 gave the best results, which was an indication that population sizes between those sizes may potentially be statistically even better. The population sizes studied were therefore extended to a population size of 800, since a population size of 1000 solutions was statistically significantly fitter in more cases than when using 500 as the population size.
   It was observed that the crossover operators performed better for high population sizes as expected, being consistently better for the largest population sizes evaluated. Given that a higher population size means a higher running time, a further population size of only 2000 was considered for the crossover operators. If there are too few solutions in a population and given that crossover used the information in the parent solutions, then the operator explores only a small part of the search space. On the other hand, if there are too many chromosomes, the algorithm may slow down, as some operations are applied to the full population.

The summary of overall results, when compared using the Mann–Whitney test, is shown in Table 7, where italics is used for the values close to those that provided the overall statistically significantly fitter solutions which are presented in black.

With respect to the mutation operators, which are based on a local search, the solutions reached are highly dependent on the individual parent solution, which generally represent small populations. Given that mutation operators require only a parent solution, the population size could range from one solution to many. As the smaller population size would consist of one solution, it may be considered that a population size of one should be the best approach from a mutation operator point of view. This relies strongly on the quality of the solution in reaching either a better or optimal solu-

Table 7 Summary of the results of the Mann–Whitney test for significance level of 0.05, different population sizes and replacement strategies

Operator	3-Pier topology		4-Pier topology	
	Population size	Selector	Population size	Selector
194 flights (16 December 2009)				
C1P	2000	IS1SUMS, *IS1ES*	2000	IS1SUMS, *IS1ES*
C2P	2000	IS1ES, *IS1SUMS*	2000	IS1SUMS, *IS1ES*
DSEMO	800, *1000*	IS1ES	500, *800, 1000*	IS1ES
MEBPFNR3	10, *5*	IS1ES	5, 15	IS1ES
MEFNR3	1, 5	IS1ES	10	IS1ES
RMEFNR2	15	IS1ES	10, *15*	IS1ES
163 flights (1 March 2010)				
C1P	2000	IS1SUMS, *IS1ES*	2000	IS1ES, IS1SUMS
C2P	2000	IS1ES, IS1SUMS	2000	IS1ES, IS1SUMS
DSEMO	1000, *800*	IS1ES	500	IS1ES
MEBPFNR3	*5, 10*	IS1ES	*5, 10*	IS1ES
MEFNR3	10	IS1ES	10, *5*	IS1ES
RMEFNR2	15	IS1ES	15, *10*	IS1ES

tion, as the fitness does not normally give a clear indication of the solution quality with respect to better solutions in its neighbourhood, which the empirical results corroborate. A solution with lower fitness may be closer to a better or optimal solution for the moves performed by the operators used, thus improving the chances that these solutions latter are reached.

In general, crossover operators are expected to benefit from large population sizes, which is corroborated by my results. Given that the crossover operators take advantage of good differences between the parent solutions, the minimum population size required is two solutions. This is the main factor benefiting crossover operators since a large population size normally results in greater diversity within the population of solutions. Nevertheless, a higher population size also means a slower algorithm execution time, given that some operations are executed for all members of the population, the processing time of which depends on the number of solutions in the population. Additionally, too much diversity may result in a loss of solutions with good building blocks, and have a corresponding detrimental effect on the overall search, the loss of better solutions or the opportunity to reach these better or optimal solutions.

As observed, the population size and operator have an important impact on the algorithm's performance, but it is not the only factor to consider, as the diversity may also be increased or decreased by changing the selection approaches used, i.e. Replacement Strategies (Sect. 1) and the Parent Selector (Sect. 2). Elitist Sampling (ES) reduces the diversity, as it keeps the solutions with higher fitness, which tends in turn to concentrate the solutions around those with fewer differences, but increases the pressure, whereas Stochastic Universal Modified Sampling (SUMS, Sect. 5) increases the chance of solutions with lower fitness taking part in the population of solutions so increasing the diversity. To reduce the ES potential detrimental effect, the Index Selector (ISxy, Sect. 5) was designed, implemented and run, the empirical results of which show a better performance than the underlying Replacement Strategies used, such as ES and SUMS.

8.2.1 Population size for when combined operators are used

Where different operators have a preference for different population sizes, these results may be taken into account when combining operators in order to improve the performance. So when the operator is selected from a pool of operators, randomly for example, its population size preference should be borne in mind so that the parent(s) may be selected within the solutions in the population and within that given preferred size. This assumes that the solutions are ordered in some way. This approach allows better solutions obtained by the other operators with larger preferred population sizes to enter the population of the current operator, potentially increasing the diversity, which it could be considered as a type of migration. In this approach, only the preferred population size is used to select the parent(s) for a given operator.

8.2.2 Run-time results for the different population sizes

In this section, the y-axis of the graphics is the average execution time for each set of 30 experiments with a different number of BSSs (the number of BSSs is shown in the x-axis). Each graph shows the average results for a given operator and

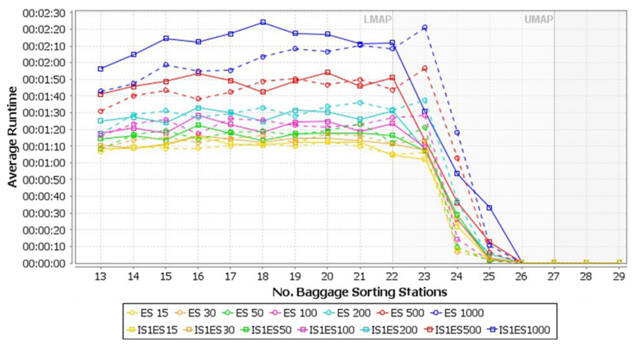

Fig. 15 Average run-time for a 4-pier topology, 194 flights (H1T091216), DSEMO and different population sizes

data set, taken from those data sets provided by NATS Ltd., different replacement strategies (ES and IS1ES) and population sizes. The lines within a graphic identify the set of experiments which were run with the same parameters, i.e. replacement strategy, population size, operator and data set.

The average results for the operator DSEMO and the different data sets are presented in Fig. 15, which shows that DSEMO requires a more constant running time up to the vicinity of the UMAP, where the running time drops to zero. On inspection of the initial population of solutions, it is apparent that the average running time of near zero refers to all the instances where the initial solutions have full assignment of flights to BSSs. So the DSEMO is unable to exchange or increase the flight assignments. As the number of BSSs is reduced up to LMAP more flights are unassigned in the initial solutions, which in turn gives the operator more chance to improve the solutions by increasing the number of assignments, potentially generating solutions with full assignments, so improving on the fitness. Finally, for numbers of BSSs lower than LMAP, not all the flights can be assigned to BSSs, so the operator initially has a chance of increasing the number of assignments for those initial solutions which do not have maximum assignment. This may also improve on the other objectives by exchanging unassigned flights with assigned ones, as will be seen in the following sections. This explains the relatively constant average running time, as the majority of operations are exchanges between assigned and unassigned flights, whereas the small variations in running time are a consequence of the number of solutions without maximum assignments in the initial solution and the speed with which the replacement strategy removes them. The differences between the various lines in Fig. 15 correspond to different population sizes, so a higher population size results in higher running times as may be expected: this is mainly because other operations are performed on all of the population members, such as applying the Replacement Strategies and the Member Selector. The difference between lines for the same population size and different replacement strategies are an indication of how quickly the replacement strategy manages to remove solutions with low fitness, i.e. those solutions which do not have maximum assignment, such as those introduced as initial solutions. This is corroborated by the fact that ES has smaller average running times than IS1ES, as expected, since ES provides a higher search pressure giving less chance for solutions of a lower fitness to generate new solutions. Also as expected, data sets with a higher number of flights required longer running times. These results also corroborate the findings presented in Ascó et al. (2012).

Figure 16 shows that the Multi-Exchange Mutation Operators (MEMOs) have a tendency to increase the running time as N (number of BSSs) increases, which corresponds to an increase in the maximum number of flights assignable and the number of initial solutions which have full assignment. Conversely, RMEFNR2 running time is near constant in most of the instances. RMEFNR2 running time for IS1ES does not appear to be affected by the number of BSSs, whereas for MEBPFNR3 and MEFNR3 the running time increases as the number of BSSs increases. Similar results were obtained for the data set provided by NATS Ltd. for 1 March 2010 and both 3-pier and 4-pier topologies.

Figure 17 shows a considerable difference in behaviour between C1P and C2P as the number of BSSs increases, whereas with C1P the speed fluctuates around an average, and for C2P the speed reduces with minor fluctuations overall according to the number of BSSs.

The mutation operators considered are much faster than the crossover operators as is to be expected. C2P and DSEMO present variations depending on the number of BSSs, whereas C2P expends more time running with very low numbers of BSSs. This is reduced as the number of BSSs increases up to a point just before the LMAP, where the required running time is kept at its lowest and most constant, irrespective of the number of BSSs.

In all the cases, as the population size increases so the running time also increases as shown in Figs. 15, 16 and 17. Similar results were obtained for the data set from London Heathrow airport Terminal 1 for 1 March 2010.

8.3 Number of iterations in a generation

The SSEA is composed of ℓ iterations per generation which contributes to the overall performance of the algorithm. Having an idea of the effects and contributions of this parameter will help in tuning the algorithm. To this end, multiple experiments were conducted using different values of ℓ for the different parameters presented below.

1. The operators used: C1P, C2P, DSEMO, MEBPFNR3, MEFNR3 and RMEFNR2.
2. Population sizes used: 1000 for C1P, C2P and DSEMO and 15 for the Multi-Exchange Mutation Operators.
3. Replacement strategies used: ES and IS1ES.

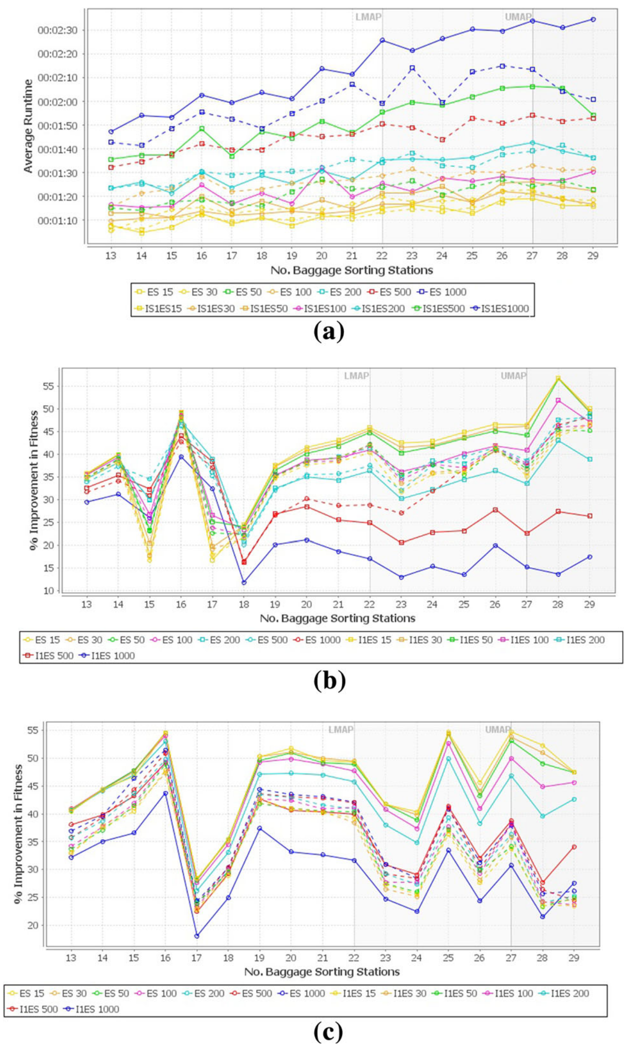

Fig. 16 Average run-time for a 4-pier topology, 194 flights (H1T091216) for some mutation operators and different population sizes. **a** MEBPFNR3, **b** MEFNR3 and **c** RMEFNR2

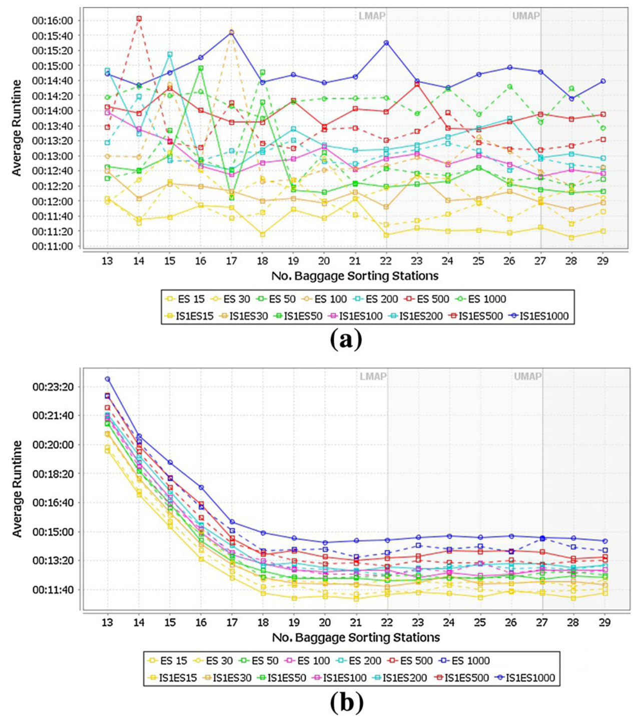

Fig. 17 Average run-time for a 4-pier topology, 194 flights (H1T091216), crossover and different population sizes. **a** 1-point crossover (C1P) and **b** 2-point crossover (C2P)

4. Initial solutions were obtained by running the constructive algorithms presented in Ascó et al. (2010, 2011).
5. Iterations in a generation: $\ell \in \{1, 5, 10, 15, 20, 30, 100\}$.
6. The data sets used correspond to those provided by NATS Ltd. (NATS Ltd.) both for H1T091216 and for H1T100301, with both 3-pier and 4-pier topologies.

The increase of ℓ equates to a reduction in the search pressure given that the current solutions have more chance of being selected as ℓ increases. Also since the same population exists for longer (ℓ times), the diversity is retained for longer as ℓ increases, e.g. if SSEA is run with a population size of 1000 solutions, for 1000 overall iterations and $\ell = 1000$, then the initial population will be maintained throughout the whole execution.

These results are similar for the different data sets and topologies considered, an overall summary of which is presented in Table 8. Table 8 summarises the values of ℓ, which provide statistically significantly fitter solutions for the widest range of numbers of BSSs. The values for ℓ between brackets are the next best values of ℓ.

Table 8 shows that C1P, C2P and DSEMO provide statistically significantly fitter solutions for all data sets considered, topologies and number of BSSs for $\ell = 1$, whereas the remaining operators considered provide statistically significantly fitter solutions in the range of ℓ from 5 to 30.

Increasing ℓ gives more chance for other solutions to be selected to generate new solutions, which equates to a reduction in pressure (but not an increase in diversity). Given that C1P, C2P and DSEMO have a large population size of 800, 1000 and 2000 solutions, respectively, which provide diversity, the same cannot be said about search pressure, which may be said to explain the preference for low values of ℓ. This also seems to be corroborated by the results for MEBPFNR3, MEFNR3 and RMEFNR2, which prefer higher values of ℓ.

8.4 Index for ISxES

The initial results obtained from the Index Selector were for a group size of $x = 1$ for the Elitist Selector (ISxES, Sect. 5) and provided solutions with good fitness. Other experiments were conducted to see what other values for x could achieve.

Table 8 Overall summary of the best ℓ of each operator, data set and topology considered and significance level of 0.05

Operator	Selector	194 flights (H1T091216)		163 flights (H1T100301)	
		Topologies			
		3-Pier	4-Pier	3-Pier	4-Pier
C1P	IS1ES	1	1	1	1
C2P	IS1ES	1	1	1	1
DSEMO	IS1ES	1	1	1	1
MEBPFNR3	IS1ES	1, 15 *(20, 30)*	5, 15, 30	10, 15 *(30)*	5, 30 *(1, 20)*
MEFNR3	IS1ES	15 *(5, 20)*	10 *(30)*	20, 100	10, 100 *(20)*
RMEFNR2	IS1ES	10 *(20, 5, 15)*	5 *(1)*	10 *(1, 5, 15)*	5 *(1)*

The characteristics of the selector indicate that any index must be greater than zero as a maximum group size of zero does not have any meaning. Moreover, there is no significance in having an index higher than the population size, since the maximum size of a group cannot be larger than the population size. Taking these factors into account together with the previous results in which the Multi-Exchange operators provide statistically significantly fitter solutions for population sizes of 5, 10 and 15, some experiments were then designed to examine the effect of changing the index $x \in \{1, 2, 3, 5, 10, 15\}$ for a population size of 15 for the Multi-Exchange Operators. Given that the preferred population sizes for the crossover operators and DSEMO are high (around 1000 solutions), a population size of 1000 was used for these operators.

The figures used in this section show the experiment results for the maximum sizes of differing group when using some of the operators previously presented. The results are presented as an average percentage improvement in fitness (y-axis), with 0% referring to the best initial solutions used. 100% refers to the upper bound obtained when running CPLEX solver with the Integer Linear Programming (ILP), for different numbers of BSSs (x-axis), Eq. 10. Negative percentages refer to the best final solutions which have a worse fitness than the best initial solution.

$$\%\text{Improvement Fitness} = \frac{f - f^{\text{Ini}}_{\text{Best}}}{f^{\text{CPLEX}}_{\text{UB}} - f^{\text{Ini}}_{\text{Best}}} * 100 \qquad (10)$$

Figures 18 and 19 show the results for the data set H1T091216, different operators and a 4-pier topology. Similar results were obtained for a 3-pier topology and the data set H1T100301.

The initial inspection of the way the operator works suggests that an increase in the index should correspond to a reduction in the diversity, as the overall number of different solutions will be reduced since many solutions with the same fitness are included in each group. As an illustration of this, the case of an operator with a population size of 10 and index of 10 is explored. As the execution progresses, it could at some time finish with 10 solutions having the same fitness, which corresponds to a behaviour similar to ES.

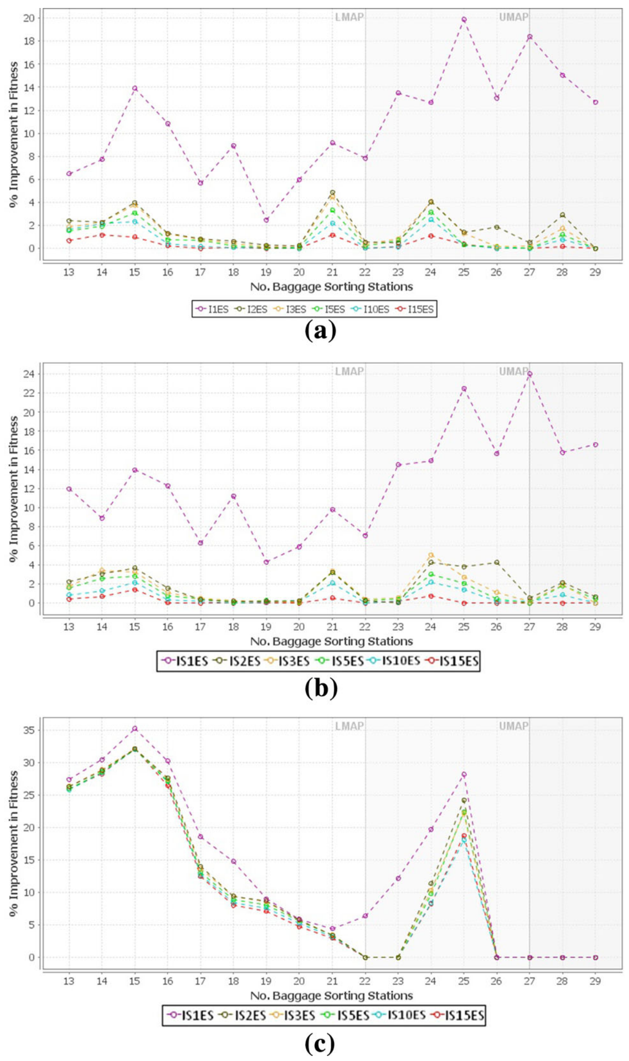

Fig. 18 ISxES $x \in \{1, 2, 3, 5, 10, 15\}$ for H1T091216 and a 4-pier topology. **a** 1-point crossover (C1P) with a 1000 population size, **b** 2-point crossover (C2P) with a 1000 population size and **c** DSEMO with 1000 population size

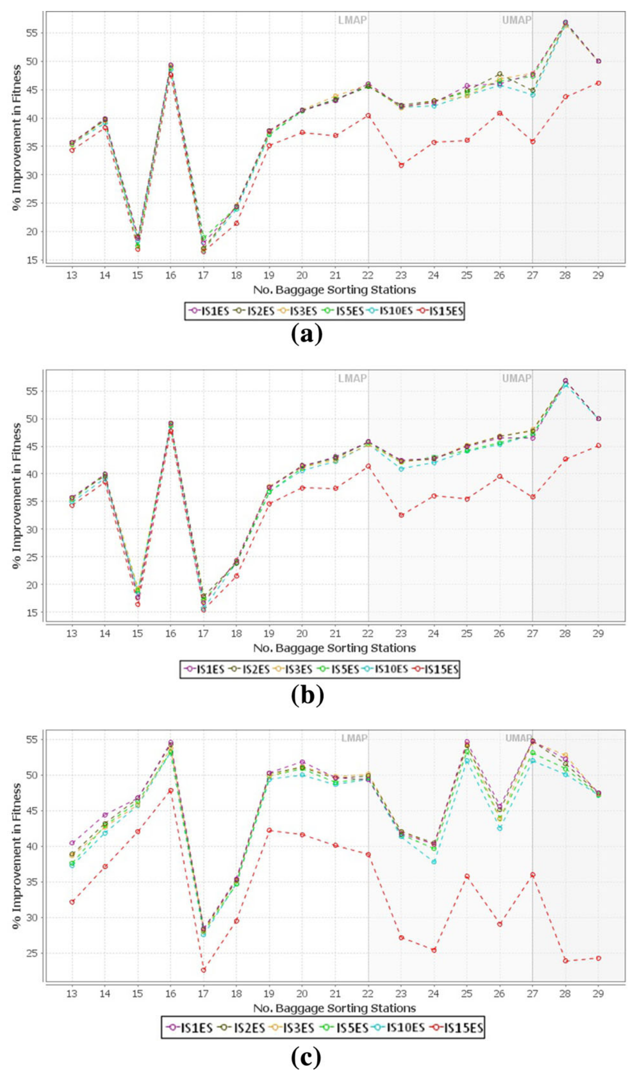

Fig. 19 ISxES $x \in \{1, 2, 3, 5, 10, 15\}$, mutation operators for H1T091216 and a 4-pier topology. **a** MEBPFNR3 with a 15 population size, **b** MEFNR3 with a 15 population size and **c** RMEFNR2 with a 15 population size

The normality test showed that it was not possible to assume that the distributions are normal; thus, it was appropriate to use the Mann–Whitney statistical significance test for each of the number of BSSs considered and between the different operators and indexes. A summary of the results for these experiments is shown in Table 9, which shows the maximum group sizes (x) only, which provided statistically significantly fitter solutions.

In general, IS1ES provided more instances with statistically significantly fitter solutions than IS2ES, IS3ES, IS5ES, IS10ES and IS15ES. In cases where both IS2ES and IS1ES perform well, IS2ES was considered better because in the cases where it provided statistical significantly fitter solutions these corresponded to a high number of BSSs, which incidentally also corresponds to the range of numbers of BSSs normally operating at an airport.

8.5 Single operators

Several experiments were run to establish the performance of each of the operators considered individually when used with the proposed SSEA. Following the previous results, new experiments were designed to establish an appropriate combination for use of an operator and replacement strategy. The parameters used in the experiments are:

1. The data sets used correspond to those provided by NATS Ltd. both for H1T091216 and for H1T100301, with both 3-pier and 4-pier topologies.
2. Number of BSSs of $N \in [13 \ldots 29]$.
3. Initial solutions were obtained by running the constructive algorithms presented in Ascó et al. (2010, 2011), as in previous sections.
4. Operators used: MEBPFNRn, MEFNRn and RMEFNRn with $n \in [2 \ldots 10]$. Also MEBPRNRn, MERNRn and RMERNRn with $n = 10$ were studied.
5. Population sizes used: 30.
6. Iterations in a generation used: $\ell = 1$.
7. Replacement strategies used: ES, IS1ES, SUMS and IS1SUMS.

Table 9 Overall summary of the Mann–Whitney statistical significance tests for index in ISxES and significance level of 0.05

Operator	194 flights (H1T091216)		163 flights (H1T100301)	
	Topologies			
	3-Pier	4-Pier	3-Pier	4-Pier
C1P	IS1ES	IS1ES	IS1ES	IS1ES
C2P	IS1ES	IS1ES	IS1ES	IS1ES
DSEMO	IS1ES	IS1ES	IS1ES	IS1ES
MEBPFNR3	IS1ES and IS2ES	IS1ES	IS1ES	IS1ES and IS2ES
MEFNR3	IS1ES	IS2ES	IS2ES	IS2ES
RMEFNR2	IS1ES	IS1ES	IS1ES	IS1ES

Table 10 Summary for SSEA1 with a single operator, 30 population size, 800,000 iterations, a 4-pier topology for 194 flights (H1T091216) and a significance level of 0.05

Operator	Number of BSSs			
	13	14	15	16
MEBPFNR3			IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEBPFNR10	IS1ES			
MEBPRNR10	IS1ES	IS1SUMS	IS1SUMS	IS1ES and IS1SUMS
MEFNR4		IS1SUMS	IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR5	IS1ES	IS1ES	IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR6	IS1ES	IS1ES and IS1SUMS	IS1ES	IS1ES and IS1SUMS
MEFNR7	IS1ES	IS1ES	IS1ES and IS1SUMS	IS1EA and IS1SUMS
MEFNR8	IS1ES and IS1SUMS	IS1ES and IS1SUMS	IS1ES and IS1SUMS	IS1ES
MEFNR9	IS1SUMS	IS1ES and IS1SUMS	IS1ES	IS1ES
MEFNR10	IS1SUMS	IS1SUMS	IS1ES and IS1SUMS	

Operator	Number of BSSs			
	17	18	19	20
MEBPFNR3		IS1ES	IS1ES and IS1SUMS	IS1ES
MEBPFNR4				SUMS
MEBPFNR10				IS1ES
MEBPRNR10	IS1SUMS		IS1ES	
MEFNR2				IS1ES and IS1SUMS
MEFNR3	IS1ES and IS1SUMS			IS1ES, IS1SUMS and SUMS
MEFNR4	IS1SUMS		IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR5	IS1ES	IS1ES and IS1SUMS	IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR6	IS1ES	IS1ES	IS1ES	IS1ES and IS1SUMS
MEFNR7		IS1ES and IS1SUMS	IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR8	IS1ES and IS1SUMS	IS1ES and IS1SUMS		IS1SUMS
MERNR10	IS1ES			
RMEFNR2				IS1ES

Operator	Number of BSSs			
	21	22 (LMAP)	23	24
MEBPFNR3	IS1ES and IS1SUMS	SUMS	IS1ES	
MEBPRNR10	IS1ES	SUMS	SUMS	
MEFNR3	IS1ES and IS1SUMS	SUMS	SUMS	
MEFNR4	IS1ES and IS1SUMS		SUMS	
MEFNR5	IS1ES and IS1SUMS and SUMS	SUMS		
MEFNR6	IS1ES and IS1SUMS			
MEFNR7	IS1ES			
MEFNR8	IS1ES			
RMEFNR2	IS1ES	IS1ES, IS1SUMS and SUMS	IS1ES and SUMS	IS1ES and IS1SUMS

Operator	Number of BSSs			
	25	26	27 (UMAP)	28
RMEFNR2	IS1ES	IS1ES and IS1SUMS	IS1ES	IS1ES and SUMS

Operator	Number of BSSs
	29
RMEFNR2	IS1ES and IS1SUMS

Table 11 Summary for SSEA1 with a single operator, 30 population size, 800,000 iterations, a 4-pier topology for 163 flights (H1T100301) and a significance level of 0.05

Operator	Number of BSSs			
	13	14	15	16
DSEMO	IS1ES and IS1SUMS	IS1SUMS		
MEBPFNR10			IS1ES	IS1ES
MEBPRNR10			IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR4			IS1ES and IS1SUMS	
MEFNR5			IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR6			IS1ES and IS1SUMS	IS1ES and IS1SUMS
MEFNR7			IS1ES and IS1SUMS	IS1SUMS
MEFNR8			IS1ES	IS1ES and IS1SUMS
MEFNR9				IS1SUMS
MEFNR10			IS1ES	IS1ES and IS1SUMS

Operator	Number of BSSs			
	17	18	19 (LMAP)	20
DSEMO	IS1SUMS			
MEBPFNR3			IS1SUMS	IS1SUMS
MEBPFNR6				SUMS
MEBPFNR8				SUMS
MEBPRNR10			IS1ES	IS1SUMS and SUMS
MEFNR3			IS1SUMS	IS1ES and IS1SUMS
MEFNR4			IS1ES and IS1SUMS	IS1ES, IS1SUMS and SUMS
MEFNR5		IS1SUMS	IS1ES, IS1SUMS and SUMS	IS1ES, IS1SUMS and SUMS
MEFNR6			IS1ES, IS1SUMS and SUMS	IS1SUMS
MEFNR7			IS1SUMS	IS1ES and IS1SUMS
MEFNR8			IS1SUMS	IS1ES
MEFNR9				IS1SUMS
MERNR10				IS1ES

Operator	Number of BSSs			
	21	22	23	24
MEBPFNR2				IS1ES
MEBPFNR3	IS1ES	IS1ES	IS1ES	IS1ES and SUMS
MEBPFNR4				SUMS
MEBPFNR5	SUMS			SUMS
MEBPRNR10		IS1ES		
MEFNR2				SUMS
MEFNR3		IS1ES and IS1SUNS	IS1ES	IS1ES and SUMS
MEFNR4	IS1ES, IS1SUMS and SUMS	IS1ES and IS1SUMS	IS1SUMS	IS1ES, IS1SUMS and SUMS
MEFNR5	IS1ES and IS1SUMS	IS1ES and IS1SUMS	IS1ES	IS1ES, IS1SUMS and SUMS
MEFNR6	IS1ES and IS1SUMS			
MEFNR7	IS1SUMS			
RMEFNR2	IS1ES and IS1SUMS			
RMEFNR3				IS1ES, IS1SUMS and SUMS

Table 12 Summary for SSEA1 with a single operator, 30 population size, 800,000 iterations, a 4-pier topology for 163 flights (H1T100301) and a significance level of 0.05

Operator	Number of BSSs			
	25 (UMAP)	26	27	28
MEBPFNR2			SUMS	SUMS
MEBPFNR3	IS1ES	IS1ES and SUMS	IS1ES, IS1SUMS and SUMS	IS1ES, and SUMS
MEBPFNR4		SUMS	SUMS	SUMS
MEBPFNR5				SUMS
MEFNR2		IS1SUMS and SUMS		SUMS
MEFNR3		IS1ES	IS1ES	SUMS
MEFNR4	IS1ES	IS1SUMS		SUMS
MEFNR5				SUMS
RMEFNR2	IS1ES	IS1ES	IS1ES and IS1SUMS	IS1ES, IS1SUMS and SUMS
RMEFNR3	IS1ES	IS1ES and IS1SUMS	IS1ES	SUMS

Operator	Number of BSSs
	29
MEFNR4	SUMS
RMEFNR2	IS1ES, IS1SUMS and SUMS

Table 13 Summary for SSEA1 with a single operator, 30 population size, 800,000 iterations, a 3-pier topology for 194 flights (H1T091216) and a significance level of 0.05

Operator	Number of BSSs			
	13	14	15	16
MEBPFNR3		IS1ES		
MEBPFNR10	IS1ES	IS1ES		IS1ES and IS1SUMS
MEFNR4	IS1ES and IS1SUMS			IS1ES and IS1SUMS
MEFNR5	IS1ES and IS1SUMS	IS1SUMS		IS1ES
MEFNR6	IS1ES and Is1SUMS	IS1ES and IS1SUMS		IS1ES
MEFNR7	IS1ES and Is1SUMS	IS1ES and IS1SUMS		
MEFNR8	IS1ES and IS1SUMS	IS1ES and IS1SUMS		IS1ES and IS1SUMS
MEFNR9	IS1ES and IS1SUMS			IS1ES
MEFNR10	IS1ES	IS1ES and IS1SUMS		IS1ES
RMEFNR2			ES, IS1ES and IS1SUMS	
RMEFNR3			ES and IS1ES	
RMERNR10			ES	

Operator	Number of BSSs			
	17	18	19	20
MEBPFNR3		IS1ES		
MEBPFNR5	ES	SUMS		
MEBPFNR6	ES	SUMS		
MEBPFNR8	ES			
MEBPFNR9	ES			
MEBPFNR10	ES and IS1ES			
MEBPFNR10		IS1ES		
MEFNR4		IS1ES and IS1SUMS		
MEFNR5		IS1SUMS		
MEFNR6		IS1ES and IS1SUMS		

Table 13 continued

Operator	Number of BSSs			
	17	18	19	20
MEFNR7	ES	IS1ES and IS1SUMS		
MEFNR8		IS1ES and IS1SUMS		
MERNR10		SUMS		
MEFNR9	ES			
RMEFNR2	ES and IS1ES	IS1ES	IS1ES	IS1ES
RMEFNR3	ES and IS1ES			
RMEFNR4	ES			
RMEFNR5	ES			
RMERNR10	ES and IS1ES			

Operator	Number of BSSs			
	21	22 (LMAP)	23	24
RMEFNR2	IS1ES and SUMS	IS1ES	IS1ES	IS1ES and SUMS

Operator	Number of BSSs			
	25	26	27 (UMAP)	28
MEBPFNR2				IS1ES and IS1SUMS
MEBPFNR3				IS1ES, IS1SUMS and SUMS
MEBPFNR4				SUMS
MEFNR2				IS1ES, IS1SUMS and SUMS
MEFNR3				IS1ES, IS1SUMS and SUMS
MEFNR4				IS1ES, IS1SUMS and SUMS
MEFNR5				IS1ES, IS1SUMS and SUMS
RMEFNR2	IS1ES, IS1SUMS and SUMS	IS1ES and SUMS	IS1ES and SUMS	IS1ES, IS1SUMS and SUMS
RMEFNR3			IS1ES	IS1ES and SUMS

Once again, given that the data cannot be said to follow a normal distribution, the Mann–Whitney test was used to establish the statistical significance of the solutions' fitness. Tables 10, 11, 12, 13 and 14 show a summary of the replacement strategies for different operators which cannot be said to provide statistically significant solutions with a lower fitness than the others for a significance level of 0.05. Those operators providing statistically significantly less fit solutions than any other are not shown for simplicity and clarity. The selection operators with the highest number of statistically significantly fitter solutions than other selection operators have been underlined.

Looking at the results obtained by the operator MEBPF, a pattern can be seen where the best solution obtained throughout the studied range of BSSs is obtained for a parameter $n \in [3 \ldots 6]$. This behaviour, together with the results obtained for the operator Multi-Exchange By Pier between a Random Number of 10 Resources (MEBPRNR10), which provides similar results on average to MEBPFNRn, prompted me to consider an extension of the MEBPRNRn for a range of numbers of BSSs, instead of a maximum value only,

Table 14 Summary for SSEA1 with a single operator, 30 population size, 800,000 iterations, a 3-pier topology for 194 flights (H1T091216) and a significance level of 0.05

Operator	Number of BSSs
	29
MEBPFNR2	IS1ES, IS1SUMS and SUMS
MEBPFNR3	IS1ES, IS1SUMS and SUMS
MEBPFNR4	SUMS
MEBPFNR5	SUMS
MEBPRNR10	IS1ES and SUMS
MEFNR2	IS1ES, IS1SUMS and SUMS
MEFNR3	IS1SUMS and SUMS
MEFNR4	IS1ES, IS1SUMS and SUMS
MEFNR5	IS1SUMS and SUMS
MEFNR6	SUMS
MERNR10	SUMS
RMEFNR2	IS1ES, IS1SUMS and SUMS
RMEFNR3	IS1ES and SUMS
RMERNR10	SUMS

as in MEBPRNRn, known as MEBPRRNRxy presented in Sect. 6.

On examining the results for the DSEMO, it is apparent that for a number of BSSs greater or equal to the LMAP ($N \geq$ LMAP), in some instances the DSEMO still manages to improve the initial solutions, even where the fittest initial solutions have assigned all the flights, as can be observed for the 25 BSSs in Fig. 18c. Simply applying the DSEMO alone provides improvements up to 25% for a 4-pier topology. Examining the initial solution provided, some of these solutions do not contain full assignments, so when the DSEMO operator is applied improvement can be achieved by means of an increase in assignments, which may in future guide the search in a different direction, in order to reach better solutions than those initially provided as initial solutions. This behaviour could be advantageous where this operator is used in conjunction with others, since it could move the search into other areas of the solution space which might otherwise not be investigated if this operator were not used. To evaluate whether this is the case it is necessary to design some experiments where the capabilities of the DSEMO operator can be seen working together with other operators which do not depend on the full assignment of flights to BSSs for a solution.

The search is said to be stagnated when it is confined to a part of the solution space where there are no fitter solutions than those which have already been found. Figure 20 also gives an indication of this situation, as it presents the average time at which the last fitter solution was found for both the C1P and C2P operators and the different replacement strategies considered in the experiments conducted. The time between the last fitter solution found and that taken to complete all of the generations gives an idea as to whether the algorithm for a given operator and replacement strategy has become stagnated. In the case of 1-point and 2-point crossovers, IS1ES preserves the search pressure and diversity better than the other replacement strategies, as shown in Fig. 20. It does not merely continue to find solutions for a longer time, but these solutions are better, as shown previously.

RMEFNR2 on its own also provides fitter solutions than any of the other operators considered on their own for the normal operational range of BSSs at an airport, i.e. $N \geq$ LMAP, and the data set of H1T091216, as shown in Table 10. On the other hand, for a less dense schedule represented by the data set of H1T100301 this range is reduced to $N \geq$ UMAP, as shown in Tables 11 and 12.

8.6 Trade-off between objectives

Figure 21 shows the non-dominated solutions obtained by different runs with single operators for 27 BSSs (UMAP) for the data set H1T091216, which illustrates the trade-off

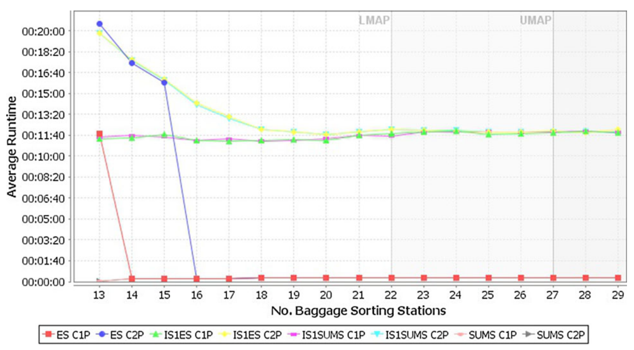

Fig. 20 Last solution found for SSEA1, 1-point and 2-point crossovers for 194 flights (H1T091216) and a 3-pier topology for 800,000 total iterations

between distance and reduction in service. It shows that the improvement in one objective corresponds to a deterioration in the other. Given that the number of BSSs is the UMAP, full assignment of all of the flights is achievable without needing to reduce the service time, which removes the need to plot the first and most important objective, the maximisation of the assignment. It should be noted that the first solution plotted corresponds to the situation where there is no reduction in service, which is possible given that 27 BSSs correspond to the UMAP.

9 Conclusions

The aim was to present the algorithm, operators, selectors and see how well the SSEA performs and to gain more general insights into the appropriate operator choices for the SSEA, especially since some operators (such as crossover) are slower to apply than others.

The SSEA, operators and selectors were presented. The empirical results for the SSEA show that this algorithm performs better than the other algorithms considered, which suggests a potential application to the problem under consideration as well as other resource assignment problems such as the AGAP.

The DSEMO extends the search to other areas of the search space which may help to improve the solutions, but it is only useful when there are unassigned flights, e.g. for $N <$ LMAP. In the case of $N \geq$ LMAP, the DSEMO should only be used when the solution selected from the population has unassigned flights, most commonly closer to the start of the search.

The different Multi-Exchange Mutation Operators presented here do not have the ability to increase the number of assignments, so for solutions which do not have maximum assignment and when maximum assignment is one of the most important objectives, these operators should not be used on their own. Given that each of the operators presented

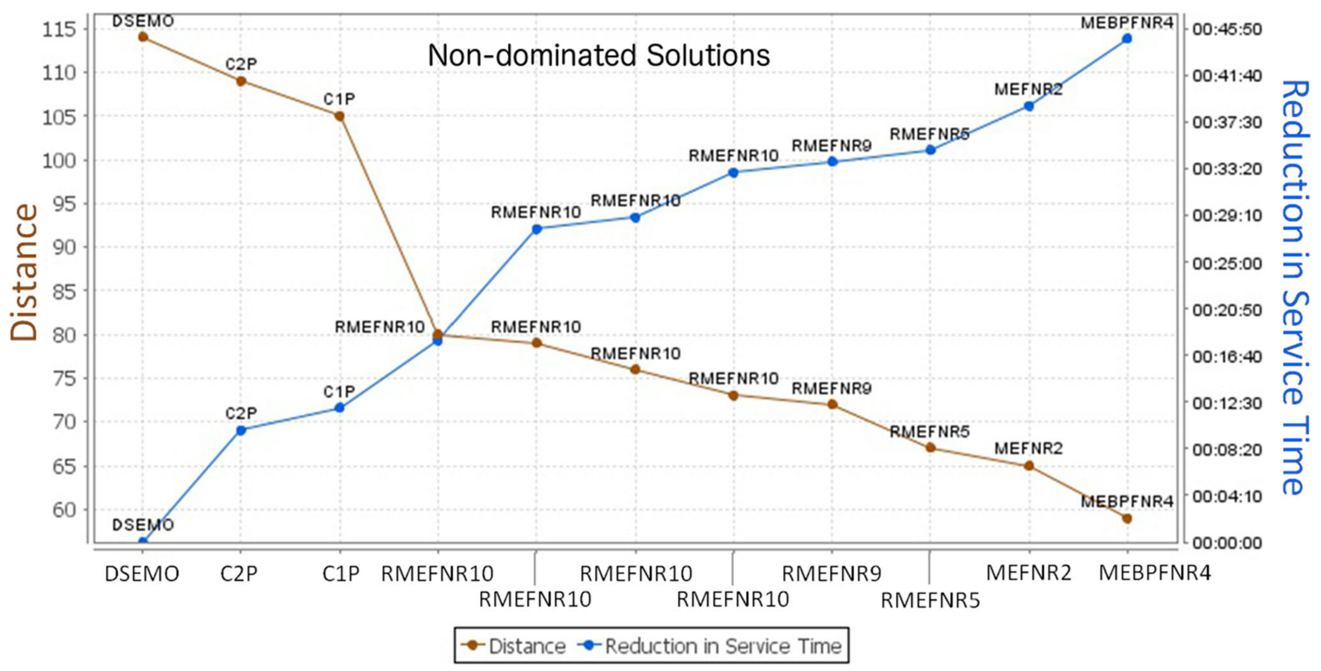

Fig. 21 Trade-offs between objectives for 4-pier topology, 194 flights, 27 BSSs and SUMS for the operators MEBPFNRn, MEBPRNR10, MEFNRn, MERNR10, RMEFNRn, RMERNR10, C1P, C2P and DSEMO with $n \in [2 \dots 10]$

has particularities, these could be used to guide the search by deciding which operators should be considered, based on the stage the search has reached at each time, e.g. if the population at a specific point in the search contains only solutions with full assignment, then the DSEMO operator should not be used.

The results presented here corroborate the importance of choosing a population size which is not only determined by the problem under consideration, but also by the operator used. The best population sizes for different operators have been shown to be very different, so there is potential for improving the performance of the algorithm when multiple operators are used by considering, for each particular operator, a sub-population of the size best suited to the operator.

Given the diverse ways in which the operators work, it is expected that their combination will further improve the solutions. Furthermore, the combination of different operators together with an adaptive method of selecting operators seems to be the most promising approach for future work. This approach could be extended to consider the number of iterations in a generation (ℓ), the value of which could be adjusted as the search progresses, to take account of the particular situation at each time.

Future work should consider extending the model to examine the capacity of each BSS, so that a more realistic number of BSSs required to service each flight can be established. The number of BSSs for each flight may initially be obtained from historical data providing the number of passengers and baggage. Furthermore, better results and robustness may be obtained if the number of BSSs required for each flight is not fixed, but depends on the capacity of the BSSs assigned to each flight and the expected checked-in baggage load each time. This means that the model not only evaluates the BSSs assigned to each flight, but also when each assignment should commence, since they may not start at the same time, thus increasing the availability of the BSSs for use in servicing other flights or absorbing disruption on the day of operation. It has been assumed that the end of the service time for all the BSS assignments to the same flight will also be the same, as it is anticipated that the volume of checked-in baggage increases as it nears the check-in desk closing time and the time for flight departures.

Acknowledgements This work was supported by the Engineering and Physical Sciences Research Council though a Doctoral Training Award, and NATS Ltd. I am grateful to the University of Nottingham for giving me the opportunity to take part in this project.

Compliance with ethical standards

Conflict of interest The author has no conflict of interest. This work was carried out by me as part of my PhD.

Ethical approval This article does not contain any studies with human participants performed by any of the authors.

References

Ascó A (2013) Constructive and evolutionary algorithms for airport baggage sorting station and gate assignment problems. Ph.D. thesis, School of Computer Science

Ascó A (2016) An analysis of robustness approaches for the airport baggage sorting station assignment problem. J Optim. https://doi.org/10.1155/2016/1213949

Ascó A, Atkin JAD, Burke EK (2010) A comparison of constructive algorithms for baggage sorting station allocation. In: Proceedings of the 24th European conference on operational research, Lisbon, Portugal. http://www.euro2010lisbon.org/

Ascó A, Atkin JAD, Burke EK (2011) Airport resource allocation using constructive algorithms. In: Proceedings of the operational research 53 annual conference. The Operational Research Society, Notthingham, UK. http://www.theorsociety.com/Pages/Conferences/OR53/OR53.aspx

Ascó A, Atkin JAD, Burke EK (2012) An evolutionary algorithm for the over-constrained airport baggage sorting station assignment problem. In: Bui L, Ong Y, Hoai N, Ishibuchi H, Suganthan P (eds) 9th Intl. conf. on simulated evolution and learning, SEAL2012, lecture notes in computer science, vol 7673. Springer, Berlin, pp 32–41. https://doi.org/10.1007/978-3-642-34859-4_4

Ascó A, Atkin J, Burke E (2013) An analysis of constructive algorithms for the airport baggage sorting station assignment problem. J Sched. https://doi.org/10.1007/s10951-013-0361-x

Baker J (1987) Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms on genetic algorithms and their application. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp 14–21

Blickle T, Thiele L (1996) A comparison of selection schemes used in evolutionary algorithms. Evol Comput 4(4):361–394. https://doi.org/10.1162/evco.1996.4.4.361

Bolender MA (2000) Scheduling and control strategies for the departure problem in air traffic control. Ph.D. thesis, University of Cincinnati ETDs. http://drc.ohiolink.edu/handle/2374.OX/12687

Burke EK, Kendall G (eds) (2005) Search methodologies introductory tutorials in optimization and decision support techniques, 2nd edn. Springer. http://www.springer.com/mathematics/applications/book/978-0-387-23460-1

Caprì S, Ignaccolo M (2004) Genetic algorithms for solving the aircraft-sequencing problem: the introduction of departures into the dynamic model. J Air Transp Manag 10(5):345–351. https://doi.org/10.1016/j.jairtraman.2004.05.004

Cheng VHL, Crawford LS, Menon PK (1999) Air traffic control using genetic search techniques. In: Proceedings of the 1999 IEEE international conference on control applications, 1999, vol 1, pp 249–254. https://doi.org/10.1109/CCA.1999.806209. http://www.cs.bham.ac.uk/~wbl/biblio/cache/http___www.optisyn.com_research_papers_papers_1999_traffic_99.pdf

Darwin C, Wallace A (1858) On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. J Proc Linn Soc Lond 3:46–50

Ding H, Lim A, Rodrigues B, Zhu Y (2004) Aircraft and gate scheduling optimization at airports. In: Proceedings of the 37th annual Hawaii international conference on system sciences, 2004, pp 1530–1605. https://doi.org/10.1109/HICSS.2004.1265219

Ding H, Rodrigues AL, Zhu Y (2005) The over-constrained airport gate assignment problem. Comput Oper Res 32(7):1867–1880

Drexla A, Nikulina Y (2008) Multicriteria airport gate assignment and pareto simulated annealing. IIE Trans 40(4):385–397. https://doi.org/10.1080/07408170701416673

Eiben AE (2003) Advances in evolutionary computing. In: Multiparent recombination in evolutionary computing. Computer science. Springer, Berlin, pp 175–192. https://doi.org/10.1007/978-3-642-18965-4_6

Eiben AE, Raué PE, Ruttkay Z (1994) Genetic algorithms with multi-parent recombination. In: Proceedings of the international conference on evolutionary computation. The third conference on parallel problem solving from nature: parallel problem solving from nature. PPSN III. Springer, London, UK, pp 78–87. http://dl.acm.org/citation.cfm?id=645822.670511

Eiben AE, Kemenade CHM, Kok JN (1995) Orgy in the computer: multi-parent reproduction in genetic algorithms. In: In Third European conference on artificial life. Springer, pp 934–945. http://www.few.vu.nl/~gusz/papers/1995-ECAL-Orgy-in-the-computer.pdf

Glover FW (1989) Tabu search—part I. ORSA J Comput 1(3):190–210

Glover F (1990) Tabusetabu—part II. ORSA J Comput 2(1):4–32

Gotteland JB, Durand N (2003) Genetic algorithms applied to airport ground traffic optimization. CEC 063:1121. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.1480&rep=rep1&type=pdf

Hansen JV (2004) Genetic search methods in air traffic control. Comput Oper Res 31(3):445–459. https://doi.org/10.1016/S0305-0548(02)00228-9

Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press. http://blog.taivo.net/post/2007/08/21/Book%3A-Adaptation-in-Natural-and-Artificial-Systems-by-John-H-Holland

Johnstone M, Creighton D, Nahavandi S (2010) Status-based routing in baggage handling systems: searching verses learning. IEEE Trans Syst Man Cybern Part C Appl Rev 40(2):189–200. https://doi.org/10.1109/TSMCC.2009.2035519

Johnstone M, Creighton D, Nahavandi S (2015) Simulation-based baggage handling system merge analysis. Simul Model Pract Theory 53(Supplement C):45–59. https://doi.org/10.1016/j.simpat.2015.01.003

Kim G, Kim J, Chae J (2017) Balancing the baggage handling performance of a check-in area shared by multiple airlines. J Air Transp Manag 58(Supplement C)(Supplement C):31–49. https://doi.org/10.1016/j.jairtraman.2016.08.017

Levinthal D, March J (1993) The myopia of learning. Strat Manag J 14(S2):95–112. https://doi.org/10.1002/smj.4250141009

Lim A, Rodrigues B, Zhu Y (2005) Airport gate scheduling with time windows. Artif Intell Rev 24(1):5–31. https://doi.org/10.1007/s10462-004-7190-4

March J (1991) Exploration and exploitation in organizational learning. Organ Sci 2(1):71–87

Mendel G (1865) Experiments in plant hybridisation. The nature research society of Brünn. http://books.google.co.uk/books?hl=en&lr=&id=3jM-mIbWUSwC&oi=fnd&pg=PA7&dq=%22Experiments+in+Plant+Hybridisation%22+Mendel&ots=RLn-N2OKZs&sig=6nJmBXxQsgs0cS-dqas1N1xV7Ns#v=onepage&q=%22Experiments%20in%20Plant%20Hybridisation%22%20Mendel&f=false

Razali NM, Wah YB (2011) Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. J Stat Model Anal 1(2):21–33

Schwefel HP, Rudolph G (1995) Contemporary evolution strategies. In: Advance in artificial life. Lecture notes in computer science, vol 929. Springer, Berlin, pp 891–907. https://doi.org/10.1007/3-540-59496-5_351

Sokolov A, Whitley D (2005) Unbiased tournament selection. In: Proceedings of the 2005 conference on genetic and evolutionary computation. GECCO ’05, ACM, New York, NY, USA, pp 1131–1138. https://doi.org/10.1145/1068009.1068198. http://www.cs.bham.ac.uk/~wbl/biblio/gecco2005/docs/p1131.pdf

Syswerda G (1989) Uniform crossover in genetic algorithms. In: Proceedings of the 3rd international conference on genetic algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp 2–9. http://portal.acm.org/citation.cfm?id=645512.657265

Tang CH, Yan S, Hou YZ (2009) A gate reassignment framework for real time flight delays. 4OR Q J Oper Res. https://doi.org/10.1007/s10288-009-0112-1. http://www.springerlink.com/content/94744705w1854316/

Tsutsui S, Jain L (1998) On the effect of multi-parents recombination in genetic algorithms. In: Second intemational conference on knowledge-based intelligent electronic systems. IEEE

Whitley D (1989) The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In: Proceedings of the third international conference on genetic algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, vol 1, pp 116–121. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.8195&rep=rep1&type=pdf

Whitley D, Kauth J (1988) Genitor: a different genetic algorithm. In: Proceedings of the rocky mountain conference on artificial intelligence, vol 1. Colorado State University, Fort Collins, CO, pp 118–130

Xiangwei M, Ping Z, Chunjin L (2010) Aircraft category based genetic algorithm for aircraft arrival sequencing and scheduling. In: Control conference (CCC), 2010 29th Chinese, pp 5188 –5193. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5573424

Yan S, Huo CM (2001) Optimization of multiple objective gate assignments. Transp Res Part A Policy Pract 35(5):413–432. https://doi.org/10.1016/S0965-8564(99)00065-8